AUTOMATED DETECTION OF HERBARIUM SPECIMENS VIA TRANSFER LEARNING IN CONVOLUTIONAL NEURAL NETWORKS

A Thesis
by
CHRISTOPHER LEIGH CAMPELL

Submitted to the Graduate School
at Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

December 2019
Department of Computer Science

AUTOMATED DETECTION OF HERBARIUM SPECIMENS VIA TRANSFER LEARNING IN CONVOLUTIONAL NEURAL NETWORKS

A Thesis
by
CHRISTOPHER LEIGH CAMPELL
December 2019

APPROVED BY:

R. Mitchell Parry, Ph.D.
Chairperson, Thesis Committee

Zack E. Murrell, Ph.D.
Member, Thesis Committee

Rahman Tashakkori, Ph.D.
Member, Thesis Committee

Rahman Tashakkori, Ph.D.
Chairperson, Department of Computer Science

Michael McKenzie, Ph.D.
Dean, Cratis D. Williams School of Graduate Studies

Abstract

AUTOMATED DETECTION OF HERBARIUM SPECIMENS VIA TRANSFER LEARNING IN CONVOLUTIONAL NEURAL NETWORKS

Christopher Leigh Campell
B.S., Appalachian State University
M.S., Appalachian State University

Chairperson: R. Mitchell Parry, Ph.D.

There are thousands of herbaria (collections of dried and mounted plants) all over the world, containing millions of specimens which have yet to be digitized and made available to online research communities. Recent global transcription efforts have utilized crowd-sourced volunteers to perform data entry, especially in areas where optical character recognition continues to fail. The costs of this, in terms of volunteer man-hours and the technical infrastructure required to operate and organize such transcription efforts on a global scale, are staggering.

Spurred by advances in computer graphics processing, the resurgence of neural networks shows promise as a viable alternative to volunteer-based manual transcription. Yet due to a variety of issues such as: the taxonomic impediment, morphologically similar specimens, and 'the species problem', automated classification in the biological domain is inherently difficult. The relatively new process of transfer learning in artificial neural networks has recently shown promise in reducing the training complexity in difficult image classification problems, despite notable differences in target tasks and domains.

Within this work, the technique of transfer learning is applied to the digital specimen collection of the I.W. Carpenter Jr. Herbarium housed at Appalachian State University, in an effort to assess its feasibility. It is shown that within the confines of the I.W. Carpenter Jr. Herbarium, the technique of transfer learning combined with modern neural networks can effectively classify specimen images to the point where volunteer-based transcriptions of certain specimen label fields may no longer be necessary. Additionally, within this work, such techniques are applied to a much larger dataset comprised of multiple international herbaria. Contrary to recently published research, it is demonstrated that such techniques may not yet be feasible on such a massive scale.

ACKNOWLEDGEMENTS

I would like to thank my parents for their efforts in supporting me both financially and emotionally throughout this long process. I would like to thank Dr. David Campell for his financial support, which enabled me to spend valuable time learning the preliminary skills that were required for this research. I would like to thank my girlfriend, Hannah Adcox, for her patience, love, and support throughout this venture. Thank you, Hannah, for bearing with me through the most difficult part of my collegiate career. I would also like to thank Dr. Tashakkori, whose enthusiasm and can-do attitude never failed to inspire. Without the referral from Dr. Tashakkori to the biology department, I may never have been given this opportunity. Thank you Dr. Tashakkori, for your role in making this possible.

Thanks goes out to Dr. Zack Murrell for providing employment in the form of a research position where I could gain familiarity with the various databases, institutions, and personnel involved in the SERNEC organization. Additional thanks to Dr. Murrell for providing valuable taxonomic expertise pertaining to the cleaning and preprocessing of class labels. Likewise, SERNEC Project Manager Michael Denslow was instrumental to the process of data obtainment. It is thanks to Michael and his willingness to share unrecorded knowledge on the SERNEC infrastructure that I was able to understand the metadata well enough to build a pipeline for data obtainment. Similarly, I would like to thank SERNEC portal manager Herrick Brown. It is Herrick who provided the

knowledge of what portions of the SERNEC portal should be targeted for metadata retrieval. Without Herrick, the data acquisition pipeline would have been needlessly more complex.

Last but not least, a tremendous debt of gratitude is owed to Dr. Parry, whose role in this project has been instrumental since its conception. It was Dr. Parry who believed in my abilities enough to take on this project as my advisor, despite knowing I had no prior experience in the machine learning frameworks required for implementation. In addition to his support, Dr. Parry has provided countless hours of advice regarding implementation techniques, debugging techniques, and development practices.

It was Dr. Parry who has spent uncountable hours explaining the difficult concepts to me, which have served to become the foundations upon which this work was built. In addition to this, Dr. Parry previously allowed me to undergo independent coursework in machine learning, at a time when such courses were both unavailable, and unpopular. Likewise, it was Dr. Parry who allowed me to be a teacher assistant for related courses, allowing me to grow my understanding of the subject beyond a superficial level. Without such opportunities to explore the material so deeply, this research would never have been possible.

When I encountered technical challenges which were miles beyond my comfort zone, Dr. Parry took the time to understand the issues I was facing, and provide valuable advice; despite having no prior knowledge of the frameworks and libraries I was utilizing. With no peer mentors with whom to confer, and no experienced machine learning engineers with whom to consult, this proved to be indispensable.

There was a time not long ago, when I used to believe that the subjects of machine

learning and artificial intelligence were simply too complex for me to understand. I feared such topics would remain forever beyond the peak of my intellectual comprehension. From your mentorship throughout this process, Dr. Parry, I have learned that no knowledge lies forever beyond one's grasp; its obtainment is only a matter of dedication, and its mastery is primarily dependent upon effort and time.

Finally, I would like to thank the SouthEast Regional Network of Expertise and Collections (SERNEC) organization, and its affiliated herbaria, academic institutions, and personnel. This publication utilizes data generated via the Zooniverse.org platform, development of which is funded by generous support, including a Global Impact Award from Google, and by a grant from the Alfred P. Sloan Foundation. Without Zooniverse's Notes from Nature platform, none of this research would have been possible on the scale that is presented within.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Significance

It is estimated that there are approximately 3,000 herbaria in the world possessing over 350,000,000 specimens in the form of mounted and dried plants [1]. Such collections possess valuable biological knowledge, but due to the physicality of their specimens, they are often isolated in localized communities of curators. It is not uncommon for specimens to be shipped between herbaria, both risking damage to the specimen and slowing the research throughput of the biologists involved. Recent digitization efforts by organizations such as iDigBio [2] have aimed to bring biological research into the modern age through the preservation and publication of digitally imaged specimens. With the advent of millions of globally accessible specimen images, massive digital transcription and annotation efforts have been underway.

Traditionally, biologists have turned to crowd-sourced transcription efforts led by volunteers who manually enter textual information captured in specimen images (e.g.

Figure 1.1). This is a slow and tedious process which produces noisy data which must then be reconciled and approved by an expert before inclusion into vetted global research databases. Furthermore, due to the high degree of manual labor such efforts require, it has been conservatively estimated that there are 70,000 species already stored in herbaria that remain un-transcribed and/or inaccessible to the global research community [3]. In an effort to address such issues, automated approaches to specimen identification have been gaining in popularity.

1.2 Motivation

Since its conception, the field of artificial intelligence has long been concerned with knowledge representation, the study of how to put knowledge into a form that a computer can reason with [4]. Knowledge representation is necessary for storing the axioms which serve as the foundation for autonomous reasoning in intelligent agents [4]. Backed by decades of work in philosophy and psychology, and originally spearheaded by interests in computational linguistics and natural language processing, symbolic knowledge representation has remained, to this day, an essential component in the design of intelligent knowledge-based agents [4]. Due in large part to its linguistic roots, symbolic knowledge representation has historically remained closely tied to the semantics of natural language. However, it is the belief of this author that it need not be a necessity that symbolic knowledge representation remain restricted by the structure of human interpretable language. The relatively new technique of transfer learning within the medium of convolutional neural networks, shows promise as a viable alternative to historical approaches of semantic-based symbolic

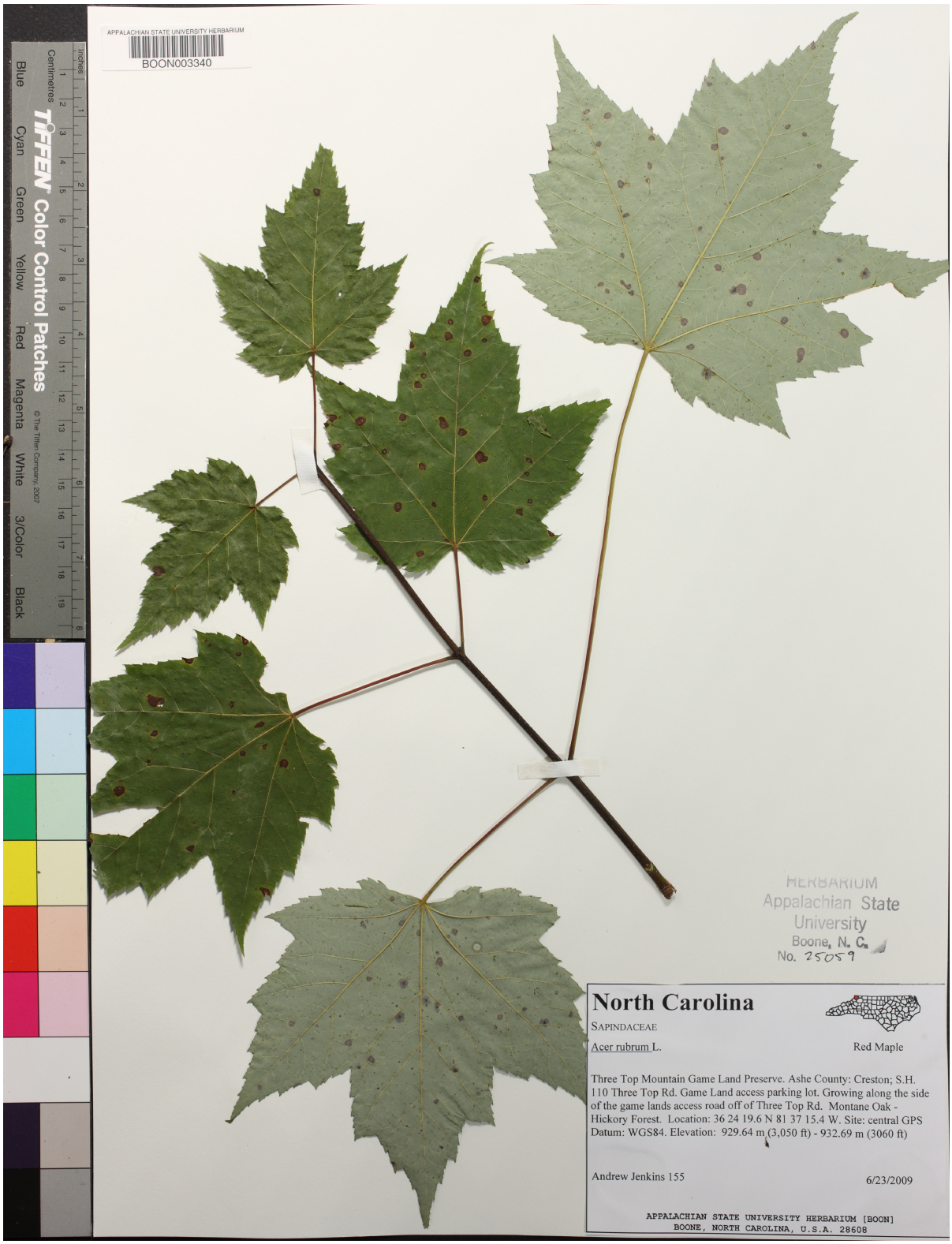

Figure 1.1: An example herbarium specimen image from the BOONE dataset with an affixed (typed) specimen label.

knowledge representation.

Often touted purely as a deep learning optimization technique, transfer learning embodies a much larger and more impactful purpose than this particular usage would suggest. Transfer learning in deep convolutional neural networks is more appropriately viewed as a particularly unique instance of representational knowledge transfer. Generalized representational knowledge transfer across domains has long been regarded as a necessary prerequisite for the development of a generalizable artificial intelligence, the 'holy grail' of the field of artificial intelligence. If transfer learning is ever to be viewed as a contender for generalized representational knowledge transfer, preliminary investigations are desperately needed. Both artificial intelligence and machine learning researchers are in need of a methodology to quantify exactly how transferable knowledge in artificial neural networks really is.

Historically, the binary label of positive transfer and negative transfer has been utilized to denote the effectiveness of applied transfer learning [5]. In its current state of the technique of transfer learning is often evaluated anecdotally, within the context of a particular implementation. Simply put, there does not currently exist a sufficient methodology to quantify if the knowledge transferred from a source task is generalizable enough to be beneficial for a new classification task in a different domain. Despite this, the machine learning based technique of transfer learning has rapidly grown in popularity for a wide range of use cases across a large number of fields, including that of automated species identification [6, 7].

Largely because there is no proposed methodology to sufficiently quantify and predict the effectiveness of knowledge transfer across domains, modern practitioners tend to

perform transfer learning with knowledge sourced from large generalizable source datasets such as ImageNet [8]. Hence, primarily through trial and error, it is known that some datasets (such as ImageNet [8]) are generalizable enough to serve as a reliable source of knowledge transfer across domains. Meanwhile, it is known that some smaller and more specific datasets are not generalizable enough to serve as a reliable source of knowledge transfer in the context of learning across largely unrelated domains. Notably, what constitutes relatedness between two target tasks is still a topic of debate [9]. Regardless, in many instances positive transfer has been observed, indicating that at least in certain applications transfer learning has provided a measurably beneficial performance gain.

This is the context that has served to inspire the research conducted herein. Automated plant species identification is a particularly difficult image classification task due to a multitude of reasons elaborated upon in greater detail in the proceeding sections. Due to the complexity of this task, it is the goal of this work to examine how feasible transfer learning is for this purpose. It is hypothesized that transfer learning may help to overcome the immense neural network training times often associated with the complexity commonly found in modern biological datasets. It is the belief of this author that if transfer learning continues to be beneficial for complex tasks such as this (across markedly different domains) then machine learning and computer vision research communities may have an increased need to construct a generalizable framework for the evaluation of such knowledge-transfer-based techniques. As a result, the beginnings of such a framework are imagined within.

1.3 Thesis Goals

1.3.1 Short-Term Goals

Examining Feasibility of Classification

As was briefly mentioned in section 1.1, the class label of 'species' (scientific name) targeted by this research is traditionally considered noisy. Due to variability among the visual features in specimen labels (see Figure 1.2), optical character recognition techniques have traditionally failed to perform adequately in this domain. As a result, the process of volunteer-based transcription is often utilized which results in noisy class labels which must then be reconciled automatically before inclusion into online databases. Recently publicized research elaborated upon within, has claimed a top-5 validation accuracy as high as 90.3%, and a top-1 accuracy as high as 79.6%, on a comparatively large herbaria dataset subjected to the same volunteer-produced labels. This is surprising considering the prevalence of irrelevant visual artifacts (see Figure 1.3) and noisy class labels. Hence, it is a goal of this work to evaluate the performance of such techniques, given the notable absence of advanced preprocessing techniques and the confounding presence of automatically reconciled class labels.

Exploration of a Candidate Hyperparameter Space

A primary goal of this research was to investigate the feasibility of transfer learning in an applied manner via the lens of automated species identification. Initial preliminary research revealed that previous studies have utilized small datasets which are not representative of the size and complexity associated with today's modern digitized biological

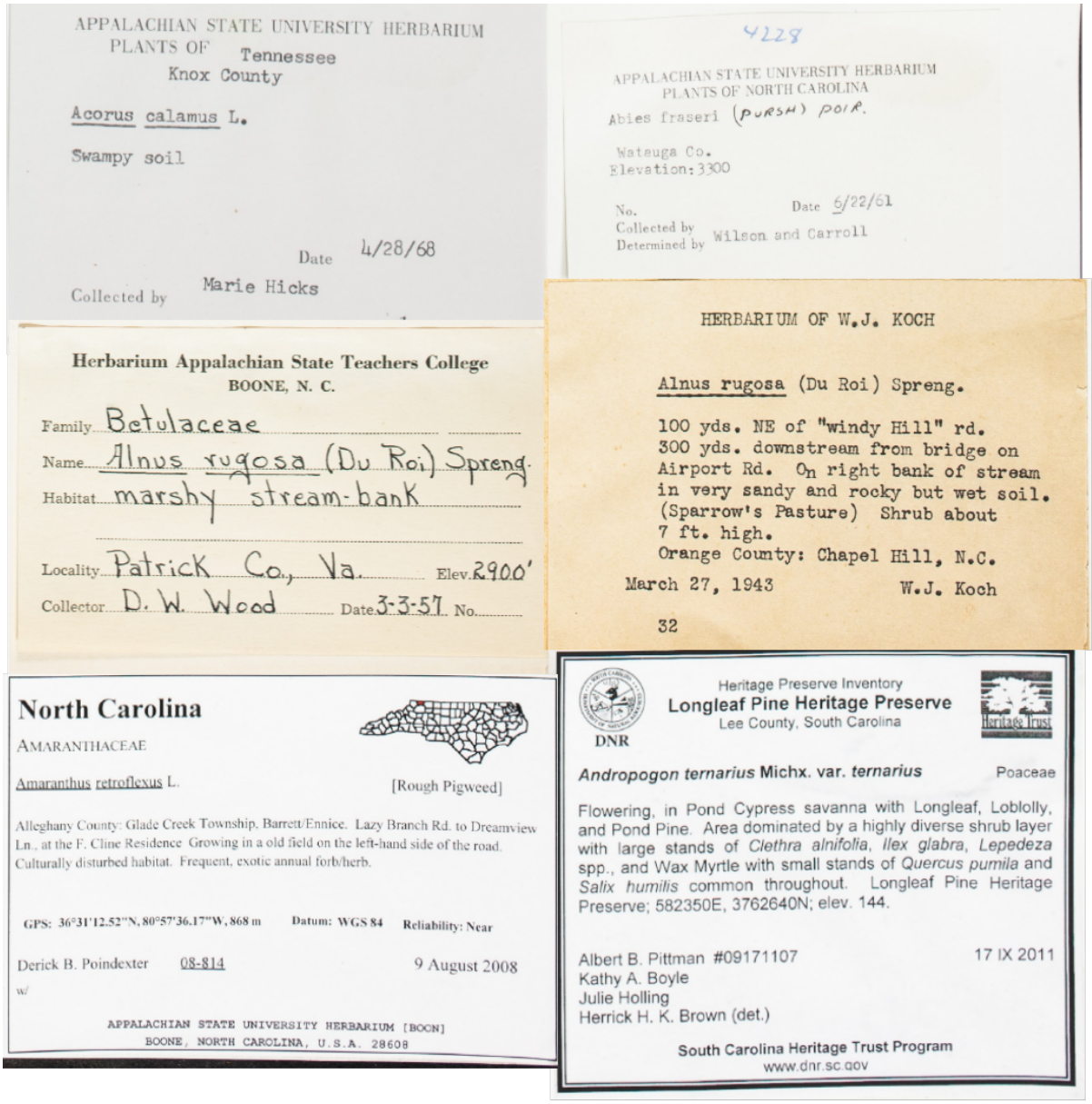

Figure 1.2: Manually selected BOONE sample images showcasing the variety of visual features among specimen labels.

Figure 1.3: Manually selected sample images from the BOONE dataset, which show examples of visual artifacts known to confuse modern neural network based computer vision algorithms.

collections. Accordingly, two distinct datasets are utilized for this research in which the size and complexity are intentionally varied to represent the growth in complexity of biological datasets. In order to assess the impact of transfer learning across both datasets it would be ideal if a common set of optimal hyperparameters was found to be shared during training. Hence, it is a goal of this work to examine if such a set of hyperparameters exists which performs well across both datasets in this domain. If such a set exists, then it is an objective of this research to determine what allows such hyperparameters to generalize well across both datasets of varying complexity. If such a hyperparameter set is found not to exist, then it is a goal of this research to develop some intuition allowing researchers to reduce the (conceivably infinite) search space of possible hyperparameters.

1.3.2 Reach Goals

Transfer Learning vs. Training from Scratch

A primary goal of this work was to assess the feasibility of transfer learning in an applied manner via the context of automated plant species identification. In practice, neural networks trained from scratch will spend time during training re-learning the generalized low-level feature detectors which have been identified to be common across many computer vision tasks. Hence, time can saved by transferring the weight values of an already pretrained neural network to a network not yet trained in a new domain (see Figure 1.4). This process of explicit knowledge transfer is known as transfer learning. When the transfer learning process results in a net-positive gain in performance when compared to an identical network that has not undergone transfer learning, then 'positive

transfer' is said to have occurred [5]. Likewise, if the net effect of the technique is a loss in performance when compared to an identical network without the transferred weight values, then 'negative transfer' is said to have occurred [5].

Previous research has acknowledged the utility of transfer learning in areas of bioinformatics, specifically during a transition from large generalized datasets to smaller less-generalized datasets [10]. Of considerably more interest to the author of this work, was the role transfer learning played in the process of artificial representational knowledge transfer across generally unrelated domains. As such, an objective of this research was to examine the impact of transfer learning between two large generalized datasets in different domains. Specifically, an objective of this research was to determine if the process of transfer learning exhibited positive, or negative transfer, within the context of knowledge transfer from ImageNet [8] for the purposes of automated plant species identification.

Fine Tuning vs. Fixed Feature Extractor

Transfer learning is often implemented in one of two ways: either by treating the source network as a fixed feature extractor (Figure 1.6), or by fine-tuning the source network (Figure 1.5). In the first case, the original pretrained neural network is left unmodified and viewed as an already developed feature extractor for use in the target domain. The original pretrained source network's weights are left unmodified, and a new (often linear) classifier is trained on the output of the original source network (see Figure 1.6). In the second instance, the pretrained source network is duplicated and the training process is continued across the entire network with new data from the target domain (see

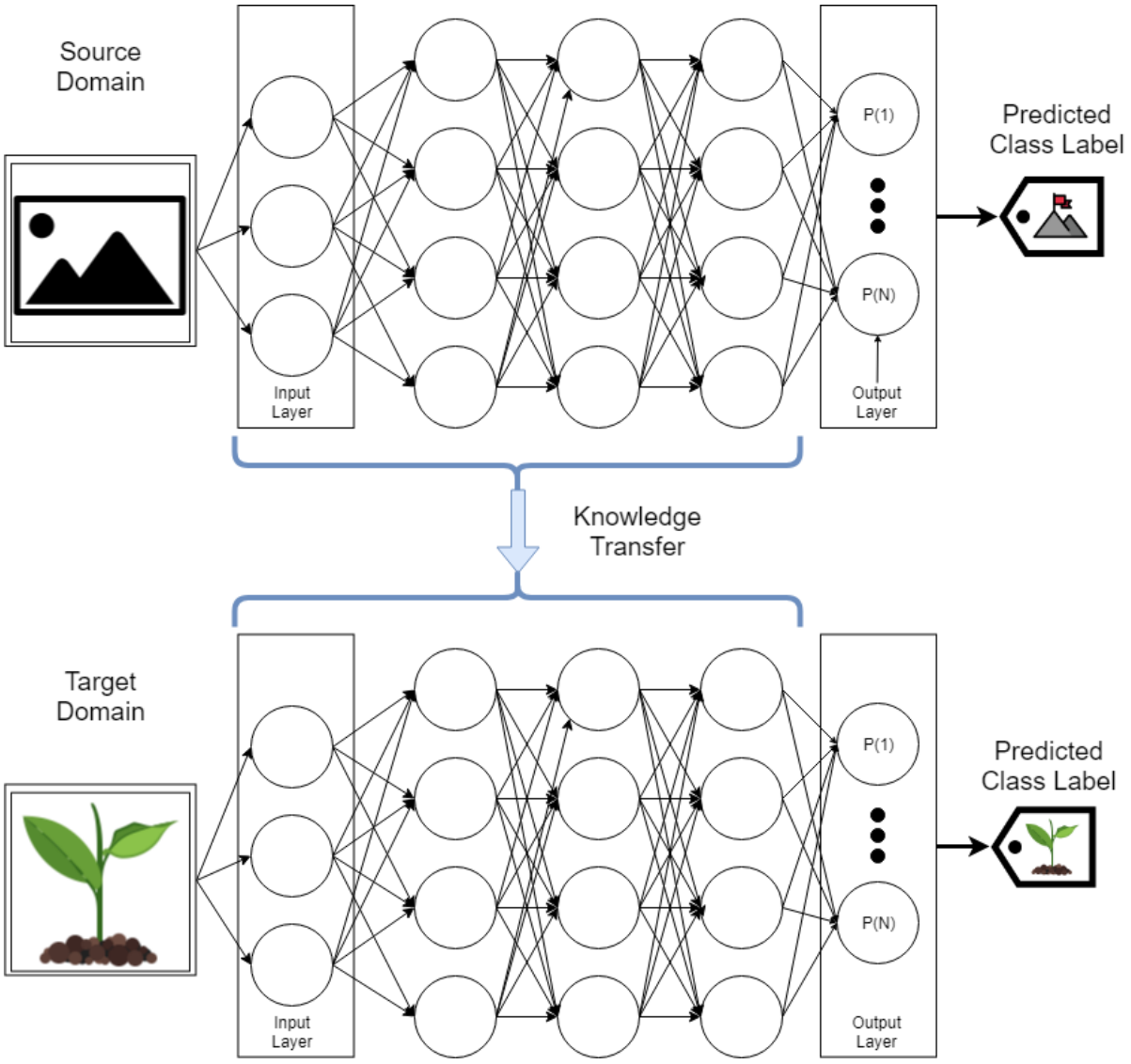

Figure 1.4: The general process of transfer learning in convolutional neural networks.

Figure 1.5). An additional objective of the research summarized within this document, was to analyze each approach.

Limits of Transferability: Frozen Layers as a Hyperparameter

When the process of fine-tuning is applied during transfer learning, the entire source network is often retrained on data from the target domain. Often a lower learning rate is employed than would otherwise have been utilized, in an effort to preserve some of the prior knowledge that is encoded in the relative values of the network's weights. Likewise, when utilizing a pretrained source network as a fixed feature extractor, it must be decided what portion of the source network should remain unmodified. The sections of the network which are to remain unmodified are said to have their weights frozen (see Figure 1.6). Often this technique is adopted when performing knowledge transfer from a large generalized dataset to a smaller more specific one. The motivating logic behind this technique is the result of the aforementioned discovery that: low-level feature detectors are developed closer to the input layers of artificial neural networks, whereas high-level feature detectors develop closer to the output layers of the network [11].

Many questions have been raised resulting from this observation and associated transfer learning technique. It is a goal of this research to provide a framework and methodology capable of answering some of the longstanding lines of inquiry surrounding this topic. Specifically, it would be immensely valuable to predict positive or negative transfer in advance. Additionally, it is hypothesized that the commonplace practice of treating the entire source network as a fixed feature extractor may not always be beneficial, considering it is known that portions of the pretrained feature extractor will

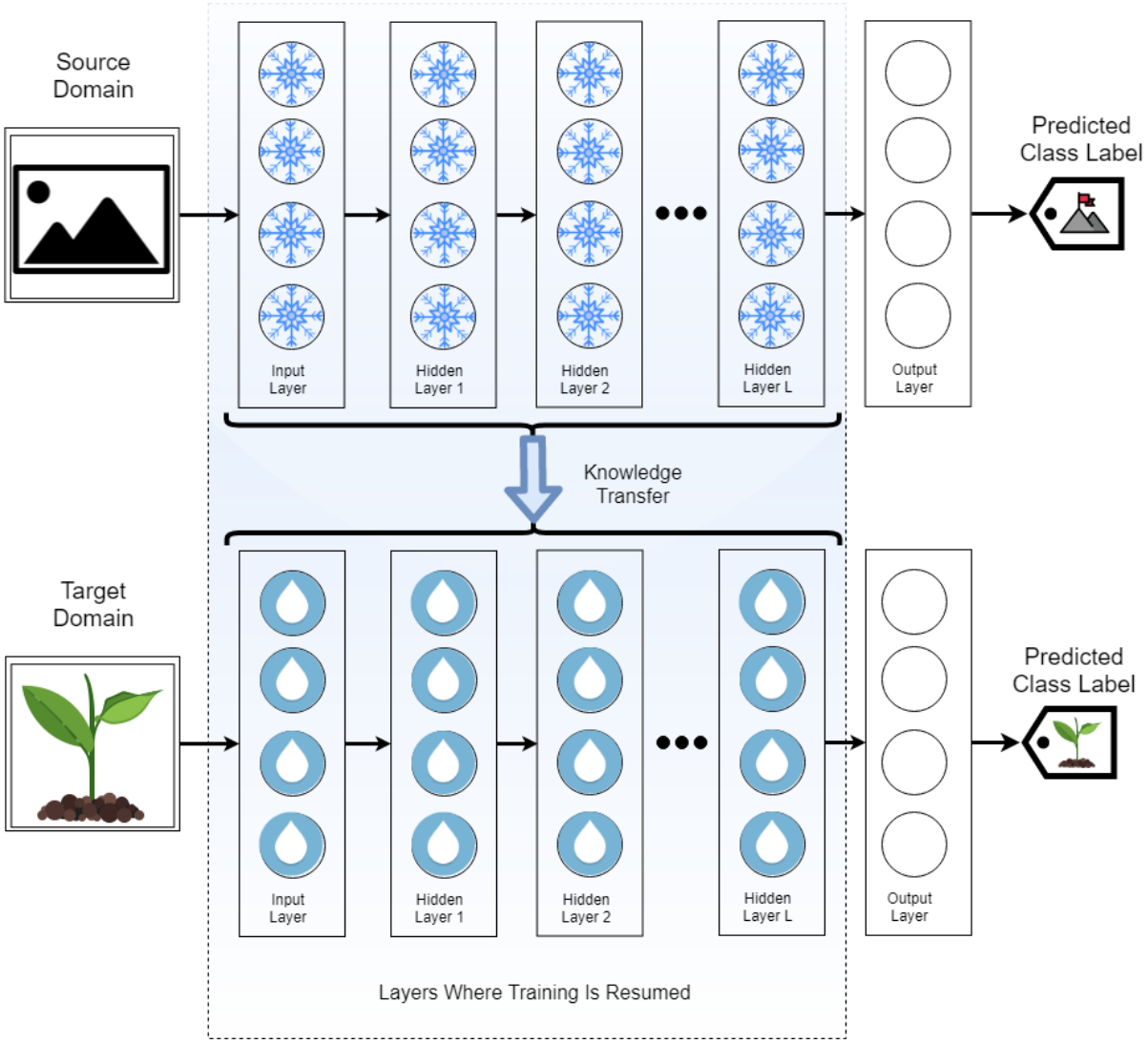

Figure 1.5: The process of fine-tuning allows for the otherwise frozen weights in the target domain to thaw, allowing them to change slightly with a significantly reduced learning rate.

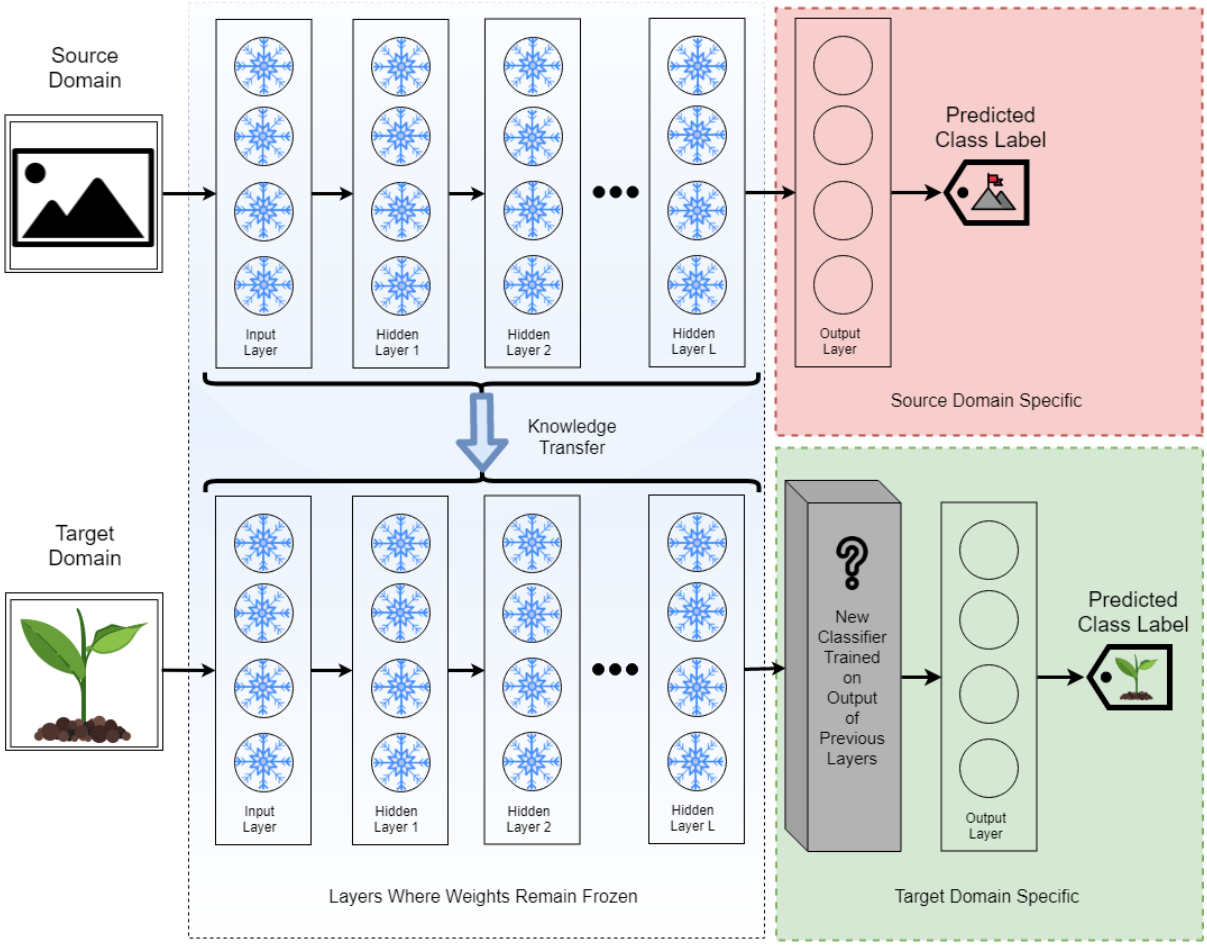

Figure 1.6: When treating the source network as a fixed feature extractor, weights remain frozen with a new (usually linear-based) classifier constructed on the output.

be domain-specific. Hence, it is a long-term goal of this work to assess the feasibility of quantifying the expected utility of cross-domain knowledge transfer in artificial neural networks, beyond just a binary label of positive or negative transfer. In treating the frozen layers of the source network as a hyperparameter to be optimized, it is hypothesized that gains in performance may be observed in the target domain by thawing portions of the source network containing lower-level feature detectors and allowing them to change.

Chapter 2

Related Work

2.1 Automated Species Identification: Why Not?

Large scale efforts by organizations such as iDigBio [2] have had the effect of bringing biological and taxonomic research into the modern age via the preservation and publication of digitally imaged herbarium specimen records. The enhanced availability of data in this sector has resulted in an increased amount of scientific publications involving the goal of automated plant species identification. In the year 2004 researchers Gaston and O'Neill published work that assessed the prevalence of automated species identification methods in the areas of pure and applied biology [12]. Gaston and O'Neill identified two primary issues that have historically withheld progress toward the development of a generic automated species identification system: the 'taxonomic impediment', and morphologically similar species [12].

2.1.1 The Taxonomic Impediment

The taxonomic impediment as described by Gaston and O'Neill, contained evidence supporting the assertion that class labels in the morphological domain are inherently subject to a high degree of noise. Relevant supporting evidence included [12]:

- A lack of an agreed upon list of species/class-labels among taxonomists
- An acknowledgement that taxology often requires frequent revision
- A notable reduction in the taxonomic work force due to proficiency requirements

Taxonomists and biologists have been unable to collectively define the term 'species', a necessary prerequisite for an agreed upon list of class labels. Collectively, the difficulties associated with the definition of the term are known as "The Species Problem" [13]. The lack of an agreed upon list of species has made it difficult to share results across biological research sub-domains. For instance, botanists may use different nomenclature than mycologists to describe the same organism. Even within the same field, botanists may use inconsistent labeling techniques. For instance, the Biodiversity Information Standard's Darwin Core Archive (DwCA) file format [14] contains 'scientificName' as a class attribute. However, the Biodiversity Information Standards official documentation allows for anywhere from one to seven sub-fields (ranging from genus to taxonomic authorship information) to constitute the DwCA attribute 'scientificName' [15]. This noisy class label has been targeted by research that has been published in well known scientific journals, without regard for a formal definition of what the classification algorithms are actually predicting.

In addition to the lack of a field-wide consensus making it hard to compare findings, a more severe implication of this is that some existing species labels have not yet been identified as synonymous with other existing labels. Taxonomic ranks are consistently in a state of flux, and species/labels are continuously refined as new genetic evidence comes to light. This violates the key assumption behind many traditional statistical and machine learning models; the assumption that training samples are statistically independent and identically distributed. With class labels in a constant state of revision, automated taxonomic classification has an additional layer of complexity not found in many other supervised image classification domains. It is this additional layer of label-associated complexity which is referred to by Gaston and O'Neill as the taxonomic impediment [12].

2.1.2 Morphologically Similar Species

In addition to the species problem, and constantly changing class labels, researchers have identified several constraints which contribute to the difficulty of the classification problem in the noisy domain targeted by this research [12]:

1. Individuals of a given species may vary significantly in morphology.
2. Closely related species may be extremely morphologically similar to one another.
3. The number of possible species to which a specimen may belong is effectively unbounded.

Much of the variation between specimens in the same species is due to factors such as age, environmental conditions, and individual genotypic variation [12]. Not only is such

variation present in biological datasets, it is actively desired. Class labels must be diverse enough to include specimens with morphological variations due to environmental conditions, yet labels must be homogeneous enough to exclude neighboring species with similar morphology [12]. Because such variability is desired, it has been deemed inadvisable to attempt to preprocess target labels/classes by removing morphological outliers. This constraint further burdened the classifier which must learn to not only distinguish between specimen images containing many similar visual features, but also among those which have undergone sometimes drastic environmental variations.

2.1.3 Statistical Methods

Despite the taxonomic impediment, morphologically similar specimens, and the species problem leading to noisy ill-defined class labels, several traditional statistical methods have managed to find some success in this challenging domain in the past. Such methods historically include [12]:

1. Principle Component Analysis (PCA) [16]
2. Linear Discriminant Analysis (LDA) [17, 18]
3. Lucas Continuous N-Tuple Nearest Neighbor Classifier (NNC) [19]

The historical success of these approaches previously led them to be worthy of consideration in this domain. However, such approaches typically ignored much of the information about the morphological structure of the specimens, instead opting to concentrate on a small subset of features. These models operated by make restrictive assumptions about

the underlying statistical distributions of the data, and have now largely been abandoned [12]. Yet such traditional statistical models have formed the foundation upon which more advanced statistical techniques have been developed. The new techniques attempted to utilize the wealth of statistical information embedded in the specimen images themselves, instead of relying on a subset of identified relevant features [12]. Arguably one of the most powerful of these modern techniques is the artificial neural network.

2.1.4 Artificial Neural Networks

Both supervised and unsupervised training of artificial neural networks have been historically utilized for the purposes of taxonomic identification with promising results [12]. In a survey of existing methods, Gaston and O'Neil highlighted the work of Boddy et al. [20] in utilizing artificial neural networks to perform taxonomic identification on phytoplankton with success rates as high as 77% [12]. Likewise, Wijesingha and Faiz developed a probabilistic neural network based classifier which reportedly obtained a testing accuracy as high as 85% on a dataset of Sri Lankan leaves [21].

Although such early results looked promising, they were conducted on relatively small datasets which did not capture the difficulties associated with the same problem today. Wijesingha and Faiz had a dataset of only 79 plant images, and only attempted to distinguish between 17 different plant species [21]. Boddy et al. obtained 77% accuracy on a dataset of only 62 class labels [20]. In stark contrast to this, Carranza-Rojas et al. utilized convolutional neural network techniques with a large herbarium test set containing 51,288 images and 1,204 different plant species; while still managing to report

an impressive 79.6% top-1 accuracy [6]. This was a noteworthy increase in size from previous datasets, and is more similar to the thousands of class labels found in today's virtual herbaria datasets.

2.2 Going Deeper in the Automated Identification of Herbarium Specimens

The work of Carranza-Rojas et al. [6] has served as a major inspiration for the research presented within this thesis. Their work attempted to significantly scale the number of species between which a classifier is able to distinguish, citing two previous studies which contained less than 100 species for classification [6]. Carranza-Rojas et al. leveraged the power of convolutional neural networks to perform automated species identification after observing the success of the technique in the published results of the 2015 International Plant-CLEF Challenge [22]. Their chosen classification model was a modified InceptionV3 architecture with batch normalization and Parametric Rectified Linear Units (PReLU) for activation functions [6].

Carranza-Rojas et al. used a batch size of 16 images during training of their modified InceptionV3 network, alongside a static learning rate of 0.0075 [6]. It is mentioned that rudimentary data augmentation was also performed with simple cropping and sample image resizing in accordance with the default settings of the popular machine learning framework Caffe [23, 6]. However, since the authors neglected to specify the version of Caffe [23] they utilized during experimentation, it proved difficult to repli-

cate the reported results. For training images, Carranza-Rojas et al. utilized the same source of data as is utilized in this thesis. Their largest dataset purportedly contained 253,733 images and 1,204 classes with an average of 207.13 images per-species in the training dataset [6]. Unlike the work presented in this thesis, Carranza-Rojas et al. used the technique of transfer learning to assess the feasibility of transfer learning between individual herbaria. In contrast, the experiments proposed in section 4.1 were concerned with the impact of transfer learning in general for this particular domain.

2.3 Plant Identification Using Deep Neural Networks via Optimization of Transfer Learning Parameters

While Carranza-Rojas et al. provided the inspiration for the work detailed in this thesis, Ghazi et al. provided the blueprint for experimentation. Ghazi et al. assessed the performance of three different deep learning architectures: GoogLeNet (InceptionV1), AlexNet, and VGGNet for the purposes of plant specimen identification [7]. The researchers also utilized transfer learning to fine-tune pretrained models, and data augmentation techniques to reduce chances of overfitting. Their published results were promising, reportedly obtaining 80% top-1 accuracy on the LifeCLEF 2015 validation set [7]. This was no small feat considering that the LifeCLEF 2015 training dataset consisted of 91,758 labeled images from approximately 1,000 species of trees, herbs and ferns [22]. Ghazi et al. utilized proportionate stratified random sampling retaining one-fifth of the samples

from each class as samples for a validation dataset [7]. Similar to Carranza-Rojas et al., Ghazi et al. utilized the popular deep learning framework Caffe [23], with source models pre-trained on the popular ImageNet subset used in ILSVRC 2012 [8].

Ghazi et al. performed fine-tuning on the pretrained models for 100,000 iterations and compared the results against the same networks when trained from scratch over the period of 200,000 epochs [7]. They set the batch size for each network to 20, with a weight decay factor and learning rate of 0.0002 and 0.001 respectively [7]. They then employed a learning rate schedule which modified the learning rate every 12,000 epochs with a decay coefficient of 0.96 [7]. Similar to Carranza-Rojas et al., Ghazi et al. concluded that fine-tuning resulted in higher validation accuracy when compared to training the identical network architectures from scratch.

Ghazi et al. also examined the computational costs associated with the number of training epochs vs the chosen training batch size. They fine-tuned pretrained models for both 100,000 epochs and 500,000 epochs, while comparing the relative training times [7]. They noted that it was reasonable to use a small but sufficient batch size with higher number of iterations. They concluded that increasing the number of training epochs had the greatest effect on trained model performance, while modifying batch size had the least effect. They specifically noted that increasing the batch size from twenty to sixty increased training time three-fold, but did not equate to the same gains in performance that can be realized from increasing the number of training epochs by the same amount [7]. Their final note was that the most significant factor affecting fine-tuning performance during transfer learning was the number of iterations, with data augmentation being the second most impactful.

Chapter 3

Data

Within this chapter, the two datasets utilized during experimentation are first introduced. Afterward, the custom data acquisition pipeline utilized to obtain the data is elaborated upon. Then, the process of class label resolution is discussed, and the data cleaning process detailed. Finally, this chapter concludes with a discussion of the Notes from Nature automated transcription reconciliation process, from which the raw class labels associated with the sample data was produced (prior to researcher obtainment).

3.1 Datasets

Two separate datasets were utilized for the purpose of this research, the BOONE dataset, and the GoingDeeper dataset. Both datasets were sourced from herbaria; as such, they were purported to contain only images of dried and mounted plant specimens. Sample images from both datasets almost always contained additional information in the form of attached hand-written or typed labels which were photographed along with the specimens

Table 3.1: A high-level view of the datasets utilized during experimentation after having been subjected to the data acquisition and input pipelines discussed in sections 3.2 and 3.3 respectively.

Cleaned Datasets Utilized		
	BOONE:	**GoingDeeper:**
Number of Classes:	88	851
Total Number of Samples:	2,791	352,938

during imaging. Each dataset was considered unbalanced, in the sense that it contained several classes with a large number of samples, and many other classes with only the minimum number of permitted samples. Both datasets possessed noisy class labels which were produced, prior to obtainment, through a process of automated reconciliation across multiple volunteer-provided transcriptions. This process of transcription reconciliation is discussed in greater detail in section 3.4.

The BOONE dataset contained every digitized and publicly available herbarium specimen record (at the time of collection) from the I.W. Carpenter, Jr. Herbarium located at Appalachian State University in Boone, North Carolina. The BOONE dataset was largely composed of plants native to the southeastern United States and the Appalachian Mountains. The GoingDeeper dataset was provided by researchers' Carranza-Rojas and Goeau et al. and was comprised of various species from multiple international herbaria, although primarily located in France [6].

3.1.1 BOONE Dataset

The BOONE dataset was a snapshot of the Appalachian State University I.W. Carpenter, Jr. Herbarium's live database of digital specimens. The dataset metadata was obtained from the herbarium's publicly accessible Darwin core archive file which was published

Figure 3.1: Randomly selected sample images from the BOONE dataset.

via the SERENC portal (http://sernecportal.org/portal/). In its final cleaned state, the BOONE dataset contained 88 unique class labels comprised of 2,791 specimen images. For reproducibility, subsection 3.2.1 describes in detail how the BOONE dataset was obtained. Likewise, section 3.3 describes the data cleaning process which all datasets underwent prior to experimentation. Table 3.2 lists the summary statistics for the BOONE dataset after having undergone data cleaning. Section 4.1 of chapter 4 describes the experiments conducted on the cleaned BOONE dataset, whereas subsection 5.3.1 of chapter 5 discusses the results of those experiments. Additional materials, including the listing of class label mappings generated during the cleaning of the BOONE dataset, can be found in the files published online with this thesis.

Table 3.2: Summary statistics for the BOONE dataset after having undergone final data cleaning.

Cleaned BOONE Dataset Summary Statistics	
Number of Unique Class Labels:	88
Total Number of Sample Images:	2,791
Enforced Minimum Number of Samples-per-Class:	20
Actual Minimum Number of Samples-per-Class:	21
Actual Maximum Number of Samples-per-Class:	82
Mean Number of Samples-per-Class:	31.72
Median Number of Samples-per-Class:	28
Mode Number of Samples-per-Class:	23 (10 counts)

3.1.2 GoingDeeper Dataset

The metadata for the GoingDeeper dataset was obtained directly from researchers Carranza-Rojas and Goeau et al. [6]. The specimen images themselves were not provided with the hosted metadata, and as such were obtained via a custom multithreaded image downloader during the data acquisition process elaborated upon in section 3.2. The GoingDeeper dataset was originally intended as a control dataset to be utilized for validating implementations of the modified InceptionV3 network architecture proposed by Carranza-Rojas et al. [6]. The metadata used to obtain the GoingDeeper dataset was both hosted as, and referred to, in the referenced research publication as the 'Herbaria1K' dataset [6]. However, the copy of the Herbaria1K dataset hosted by Carranza-Rojas et al. differed greatly in summary statistics from what was reported in the accompanying published research paper [6].

To avoid fostering the misconception that the Herbaria1K dataset discussed by Carranza-Rojas and Goeau et al. in *Going Deeper in the Automated Identification of Herbarium Specimens* [6] is identical to the actual data obtained and used in this research, a distinction in name was necessary. As such, the 'GoingDeeper' dataset is the name used

within this research to refer to the actual data obtained and utilized in the experiments described in section 4.1. Although the metadata provided by Carranza-Rojas et al. (under the name Herbaria1K) was used to obtain the GoingDeeper dataset, the two datasets do not share the same summary statistics. Unlike the 1,204 purported class labels in the Herbaria1K dataset [6], the obtained GoingDeeper dataset only possesses 851 unique classes. The GoingDeeper dataset is comprised of 352,938 samples, whereas the Herbaria1K dataset was reported to contain only 253,733 samples [6]. This significant discrepancy of 99,205 samples images has served to further motivate the distinction in name.

Table 3.3 lists the summary statistics for the GoingDeeper dataset after data cleaning was performed. For reproducibility, subsection 3.2.2 describes how the GoingDeeper dataset was obtained. Section 3.3 details the cleaning process which both datasets were subjected to prior to experimentation. Section 4.2 of chapter 4 describes the experiments conducted on the cleaned GoingDeeper dataset, whereas subsection 5.3.2 of chapter 5 discusses the results of those experiments. Similar to the BOONE dataset, additional materials related to the data cleaning process (such as class label mappings) can be found in the files published online with this thesis.

3.2 Data Acquisition

A proportionally large amount of time was spent obtaining the data necessary to perform the experiments outlined in section 4.1. For the purposes of such experiments, data was collected via two different acquisition pipelines which vary slightly in design based on the

Figure 3.2: Randomly selected sample images from the GoingDeeper dataset.

Table 3.3: Summary statistics for the cleaned GoingDeeper dataset after having undergone data cleaning.

Cleaned GoingDeeper Dataset Summary Statistics	
Number of Unique Class Labels:	851
Total Number of Sample Images:	352,938
Enforced Minimum Number of Samples-per-Class:	20
Actual Minimum Number of Samples-per-Class:	27
Actual Maximum Number of Samples-per-Class:	3,845
Mean Number of Samples-per-Class:	414.73
Median Number of Samples-per-Class:	361
Mode Number of Samples-per-Class:	313 (7 counts)

complexity of the data obtainment process. Within this section both data acquisition pipelines are described in detail for the purposes of reproducibility.

3.2.1 BOONE Data Acquisition

The BOONE dataset metadata was acquired directly from the SERNEC portal in the form of a zip-compressed Darwin Core Archive (DwCA) [14] file. The compressed BOON DwCA file was programmatically extracted producing the two relevant files:

- occurrences.csv
- images.csv

The extracted DwCA file contained a record of occurrences (occurrences.csv) and a separate (images.csv) file containing URLs pointing to specimen images on CyVerse remote servers. Both metadata files were merged into a single file (df_meta.csv) via the 'coreid' unique attribute during an SQL-like inner-merge preserving index order. Specimen records which had no 'scientificName' attribute were discarded, as well as records with no associated image URL. The remaining data was then subjected to a multi-threaded image downloader which targeted every remaining URL. Six threads (equal to the number of logical cores on the research machine) were run concurrently by the image-downloader. Images were stored in folders on the local machine according to the verbatim 'scientificName' class label. If the directory was unable to be written due to illegal characters in the 'scientificName' attribute, the record was discarded.

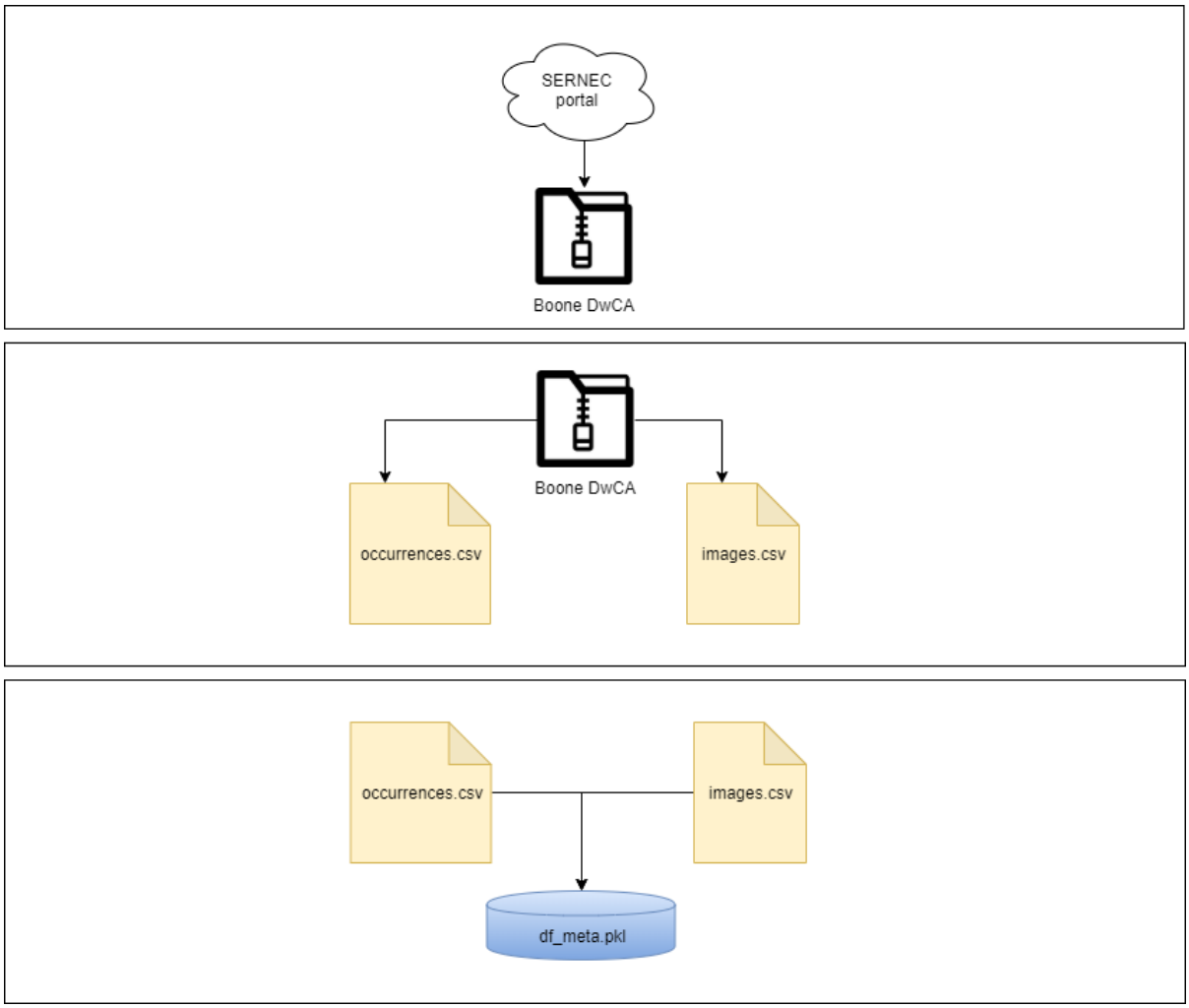

Figure 3.3: The first three stages (out of four) of the BOONE data acquisition pipeline.

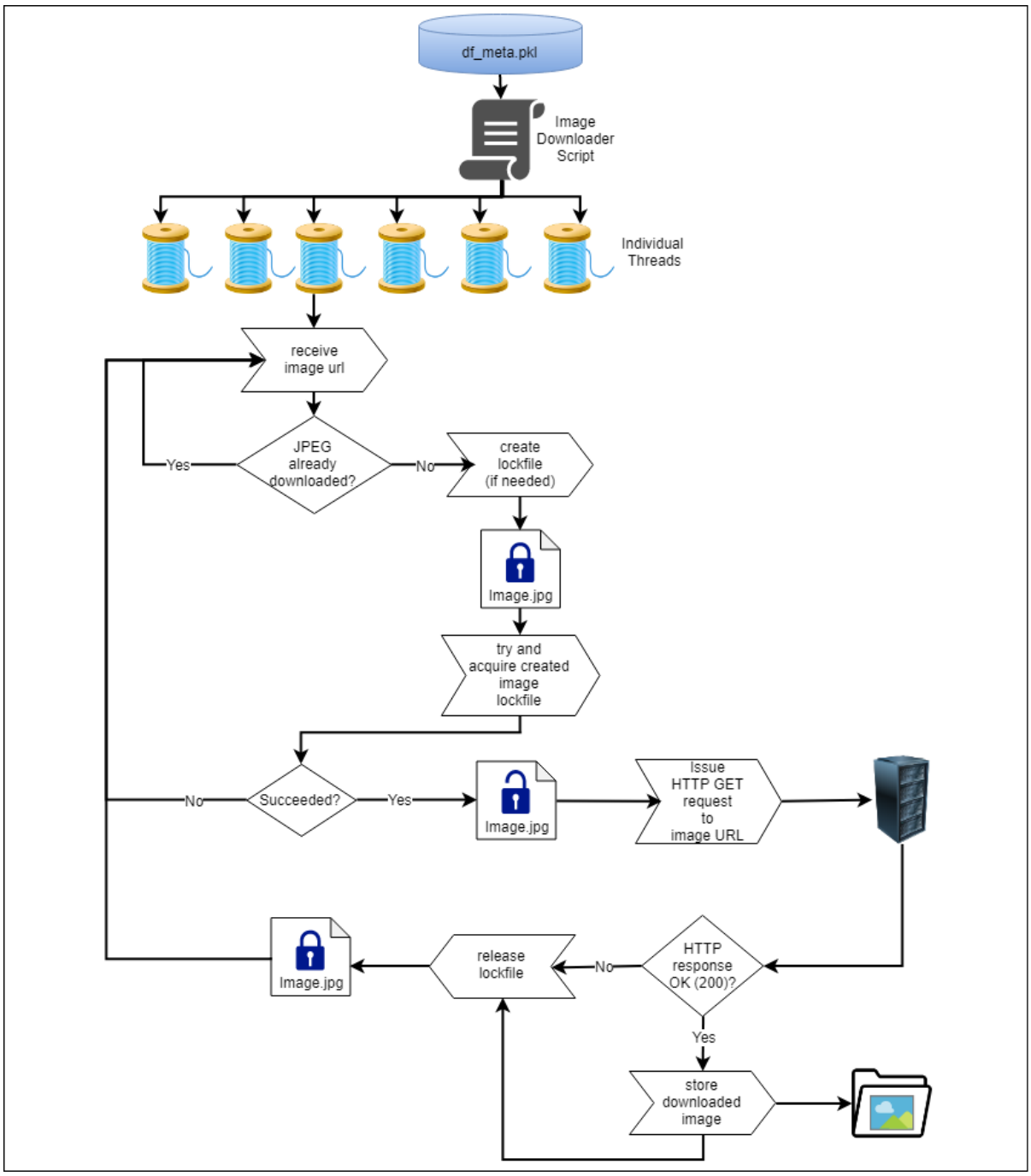

Figure 3.4: The view of a single thread in the fourth stage of the BOONE data acquisition pipeline.

3.2.2 GoingDeeper Data Acquisition

The metadata for the GoingDeeper dataset was downloaded directly from researchers Carranza-Rojas and Goeau et al.'s hosted files available at: (`http://otmedia.lirmm.fr/LifeCLEF/GoingDeeperHerbarium/`) [6]. The downloaded metadata was sourced from iDigBio [6]. As such, the same Darwin core archive attributes were utilized. At the time of obtainment, the downloaded data had already been merged into a single metadata dataframe resembling the output from stage three of the BOONE data acquisition pipeline (figure 3.3). This was used as input to a similar multi-threaded image downloader as the BOONE dataset. The process of downloading the specimen images themselves took multiple days, despite utilizing six threads concurrently. The raw downloaded images were then subjected to an additional data cleaning pipeline discussed in section 3.3.

3.3 Data Input

This section describes in detail the data input pipeline utilized on all datasets to refine and clean the acquired data prior to analysis. Although the methodology for obtaining both datasets differed slightly, it was desirable to have a single pipeline for data cleaning which could be utilized on all datasets. As a result, the data input pipeline was developed to preprocess all input data into a form suitable for experimentation. At the highest level, data ingestion is performed first via cleaning, and secondly via preprocessing. An object oriented approach was adopted to ensure uniformity throughout this process. Key responsibilities were encapsulated by two primary classes: the ImageExecutor class

and the BottleneckExecutor class. The ImageExecutor class handled data cleaning and loading. The BottleneckExecutor class performed preprocessing in both the traditional and nontraditional sense of the term. Both classes are elaborated upon in detail in the following sections.

3.3.1 Preliminary Data Cleaning Process

Prior to reaching the data input pipeline, there were a number of preliminary data cleaning operations performed previously during the data acquisition pipeline. During stage three of the pipeline, sample records containing Unicode encoding errors in target class label were flagged for removal. During stage four of the pipeline, samples containing no listed class label were removed. Additionally, samples that contained no associated image URL were removed during this stage. Then class labels flagged with Unicode encoding errors were removed. Additionally, remaining class labels with less than one hundred remaining samples in possession of image URLs were removed during this stage. As a time saving step, class names that were also invalid Window's directory names were removed. Hence, by the time the data arrived at the input pipeline, it had already undergone the preprocessing steps restated above. The purpose of the input pipeline was therefore to carry out all remaining preprocessing steps, and perform any additional prior optimization. This functionality was implemented via the two classes elaborated upon below.

3.3.2 Data Cleaning (ImageExecutor Class)

The ImageExecutor class was delegated the following tasks:

1. Obtaining raw image lists
2. Cleaning class labels
3. Ensuring the minimum number of required sample images
4. Providing protected access to image lists

The BottleneckExecutor class was coupled tightly with the ImageExecutor class. The ImageExecutor class provides the only means for the Bottleneck Executor to obtain a list of the images that are physically stored on the hard drive. The ImageExecutor ensures that when the BottleneckExecutor requests images to preprocess, the images have already been cleaned and class name resolution has been already performed.

Obtainment of Raw Images

Every time the ImageExecutor instance is instantiated it performs a walk of the root image storage directory, recording the file path of every image it encounters. Each sample image is represented in the ImageExecutor instance as a tuple, which is comprised of the image's file path and raw class label. The ImageExecutor class injects a layer of abstraction between the physical storage locations of the images on the hard drive, and the representation in computer memory. The need for this abstraction was realized after following several introductory transfer learning tutorials online. Many of these tutorials contained data already separated into distinct training and testing sub-directories. As

a result, often the tutorial's code distinguished between training and testing data based on the relative file paths in the root image storage directory.

This approach was initially adopted, until it became clear that the physical locations of files on the hard drive would need to be continuously altered as the ratio of training to testing data changed. Additionally, since the raw class names were often altered and merged during the data cleaning pipeline, the location of files would need to be continuously modified and the history of such changes lost. As a result, the ImageExecutor class provided a layer of abstraction, decoupling the physical file storage hierarchy from the representation in memory. In doing so, the ImageExecutor class had the associated responsibility of generating data cleaning reports which describe the discrepancies between the physical file storage hierarchy and the combined-and-altered class labels in memory. Importantly, this abstraction allowed the file locations of samples which belonging to raw class labels to remain unmodified on the hard drive, thereby saving hours of otherwise repeated image transfer times.

Cleaning of Class Labels

In addition to populating the list of images and raw class labels, the ImageExecutor class performed all class label resolution. During this step, raw class labels were cleaned, and aggregated when necessary. To accomplish this, the ImageExecutor class performed several stages of cleaning. First, taxonomic rank information was removed from the specimen labels. Any class label containing only the genus of the species was removed, on the grounds that morphological variations among the same genus would be too extreme, therefore violating the independent and identically distributed assumption. An example

of this cleaning step was the removal of the raw class label *Equisetum*, whereas the class label: *Equisetum Arvense* remained in the dataset since it contained a more narrow taxonomic rank than that of genus alone.

Secondly, any class label which contained a taxonomic subspecies or variety rank, had the subspecies portion of the class label removed. Hence, the scientific name: *Equisetum Arvense var. Ramulosum* was mapped to *Equisetum Arvense*. Likewise, the scientific name: *Equisetum Arvense sub. Ramulosum* was mapped to the same name: *Equisetum Arvense*. Due to the morphologically similar species problem (discussed in subsection 2.1.2), human taxonomic experts often have trouble distinguishing between subspecies within the same family. As such, it was deemed unreasonable to expect a single non-ensemble-based machine learning classifier to distinguish between subspecies with similar morphology (and therefore visual features). In order to minimize the amount of errors in the class label resolution process, all automated mappings are retained for human verification. The raw class labels and their associated mappings are included in the appendices. Additionally, statistics are exported during the process which describe the impact of the class label data cleaning on the original raw dataset.

The third step in the class label cleaning process was the removal of all taxonomic authorship information. Traditionally, the taxonomic authorship information is used to refer to the inventor of the taxonomic schema used in the labeling of the specimen image. For the purposes of the research conducted within, the author of the taxonomic naming schema is irrelevant, so long as the class labels are identical when the authorship information is removed. The removal of taxonomic authorship information allows classes that don't include this information in the scientific name field, to be correctly

interpreted as the same class by the classifier. An example of this step was the resolution of the raw class label: *barbarea verna (mill.) asch* which became: *barbarea verna ash* after the taxonomic authorship information *mill.* was removed. The removal of the taxonomic authorship information *mill.* allowed for the class *barbarea verna (mill.) asch* to be correctly identified as identical to *barbarea verna ash* independent of the taxonomic author.

The penultimate class label cleaning step performs the removal of class labels which constitute species from hybrid genera. Such species belong to multiple genera which often results in shared visual similarities among both genus. An example of this step was the removal of the raw class labels: *apocynum x floribundum* and *betula x purpusii*, both of which are hybrid genera. This cleaning step was again performed in accordance with the same rational as above: it was deemed unfair to expect a single non-ensemble based classifier to distinguish between specimens in the hybrid genera, and specimens in each of the component genera that make up the hybrid class.

The final step in data cleaning was the removal of manually identified abnormalities identified in the output from the previous cleaning steps. The list of cleaned class labels were viewed prior to this step. Class labels which were obvious misspellings of existing class labels were manually designated to be remapped during data cleaning. A list of these manually enforced class mappings can be found in the appendices. It is acknowledged that there is a degree of bias inherent in manual class label resolution. As a result of this, the manually declared class label mappings are provided for analysis and discussion. An alternative to performing this process manually would have been to utilize fuzzy string matching via the Levenshtein Distance metric [24]. However, it was

felt that performing the name resolution manually would result in less errors.

After the conclusion of the above final step, the class labels were considered to be cleaned. A dictionary of the cleaned class labels and file paths associated with the corresponding sample images was then provided to the BottleneckExecutor class for further preprocessing. Requests issued by the BottleneckExecutor class for image lists were required to go through the ImageExecutor class. The ImageExecutor class performed class label resolution once per grid search execution, and stored the cleaned image lists in memory to avoid the computational overhead associated with repeated hard drive searches.

3.3.3 Data Preprocessing (BottleneckExecutor Class)

One of the primary goals of this research was to access the feasibility of transfer learning in the domain of automated plant species identification. One of the important preprocessing steps that can be done when performing transfer learning, is the images can be forward propagated through the source network and cached on the hard drive (as shown in Figure 3.5). Then the outputted feature vectors can be loaded instead of the images themselves. This greatly speeds up computation time by avoiding a repeated forward propagation step. The purpose of the BottleneckExecutor class was to perform this operation and encapsulate any associated complexities. The end result was a compressed `pandas` DataFrame [25] containing the cleaned target label of every class, in addition to the file paths of the associated images, and their resulting bottleneck vectors.

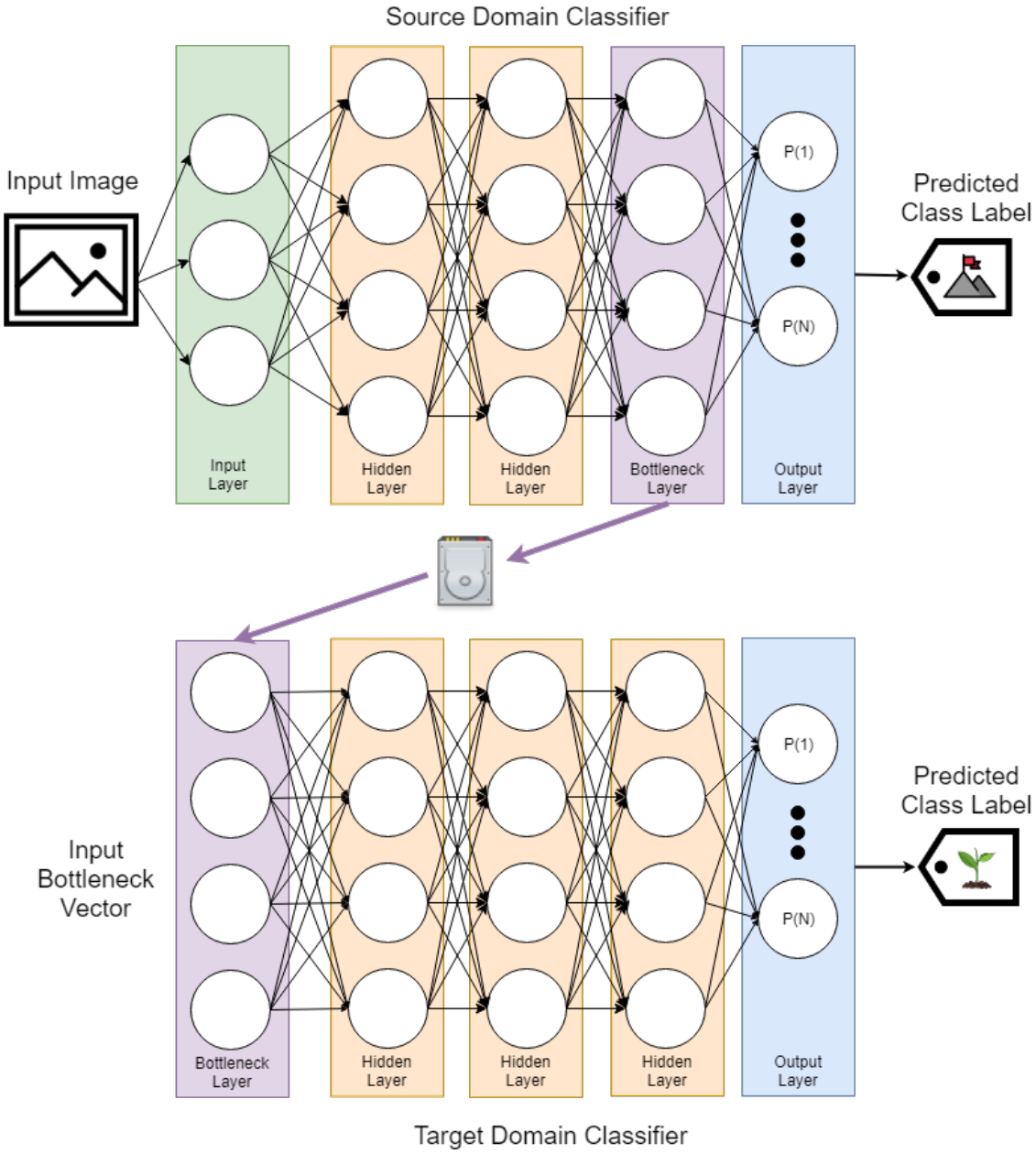

Figure 3.5: The bottleneck vector is the pen-ultimate layer of the source network. It can be pre-computed and cached to the hard drive, then used as input for a classifier in the target domain.

Data Cleaning

Data cleaning in the BottleneckExecutor class was primarily delegated to the ImageExecutor class, with few exceptions. One key exception is that the BottleneckExecutor class performed checks to ensure that TensorFlow was able to programmatically read the sample images. Prior to adding this responsibility to the BottleneckExecutor class, the training process would be sporadically interrupted by JPEG decoding errors. As a result, the BottleneckExecutor performed JPEG decoding tests on the cleaned sample images. This responsibility was given to the BottleneckExecutor class instead of the ImageExecutor class because the ImageExecutor class purposefully did not utilize TensorFlow so that it could abstain from utilizing the graphics processing unit in order to be run independently of other processes (such as data visualization). Meanwhile, the BottleneckExecutor class was required to support TensorFlow in order to generate bottleneck vectors via forward propagation. As a result, the constructed computational graph was utilized to first perform JPEG decoding error tests.

Initially, the BottleneckExecutor class was designed to forward propagate images sequentially. This was far too slow for running JPEG decoding tests. As a result, the functionality of identifying corrupted JPEG images was transferred to a separate class: 'BrokenImageLocator.py'. This was done mainly to reign in growing software complexity in the Bottleneck Executor class. Yet having the JPEG decoding tests in a separate script made it easier to experiment with techniques to increase the speed of the decoding tests without incurring a cascade of changes.

In order to process more images simultaneously, initially the BrokenImageLoca-

tor class was given the ability to attempt to decode and forward propagate as many images per batch as the GPU memory would allow for. However, a latent slowdown was incurred due to the asynchronous interaction between CPU, RAM, and GPU. Further refinement of the BrokenImageLocator class utilized TensorFlow data generator objects to pre-allocate a shuffled buffer of loaded images which were fed directly to the GPU at the rate it took to forward propagate image batches. This allowed the CPU to spend time retrieving the images during GPU processing, so that the CPU was always ready with the next batch of images as soon as the GPU finished the previous batch. This technique lead in an increase of GPU utilization from 20% to 90% utilization, and greatly sped up the throughput of the forward propagation process.

Because there were relatively few images identified to cause JPEG decoding errors, these images were removed manually by examining the list of broken images generated from the BrokenImageLocator class. However, since it was concievable that there would be more than a few broken images, functionality was incorporated into the broken image locator to address this. The broken image locator creates a blacklist of images ('blacklist.json') that the BottleneckExecutor class can receive as input and therefore ignore. Luckily, the number of problem images were so low that the implementation of the automated blacklisting was abandoned in favor of the quicker manual solution.

Bottleneck File Generation

A large amount of research time was spent designing and improving the BottleneckExecutor class. The primary responsibility of the class was to compute the bottleneck vectors for every single input image. Initial prototypes of the BottleneckExecutor performed

this process sequentially. Since performing this process sequentially took days of uninterrupted computation time, a significant effort was made to speed up the process. The sequential BottleneckExecutor took approximately 1 second per input image to forward propagate and record the resulting vector. For a dataset as large as the GoingDeeper dataset (352,938 images) this would have taken 98 hours of uninterrupted run time.

With the enhanced version of the BottleneckExecutor class, images were forward propagated at a rate of approximately 11.60 images per second. The physical restraint on this process ended up being the amount of memory available to the GPU. The memory size of the GPU placed an upper limit on the number of images that could be forward propagated per batch. Through trial and error a maximum batch size of 400 images was identified per forward propagation when utilizing the InceptionV3 network on the specified research machine. With more than 555 images sized 299 x 299 x 3 per forward pass, Out Of Memory (OOM) errors were encountered on the GPU, despite possessing 11.0 GB of dedicated GPU memory.

3.4 Automated Transcription Reconciliation Process

As was mentioned in subsection 1.3.1, the class label: 'scientificName' targeted by this research was produced, prior to obtainment, by an automated reconciliation process of digitally provided volunteer transcriptions. The Zooniverse crowd-based research website hosts the Notes from Nature platform which was used to interface with volunteer transcribers. The script used by Notes from Nature to perform automated transcription reconciliation is available online (`https://github.com/juliema/label_reconciliations/`

blob/master/README.md). As described in the associated documentation, there are two different approaches for reconciliation depending on if the field to be reconciled is a free-text entry field, or a constrained pre-populated dropdown menu.

For the transcribed fields which contain controlled vocabulary (e.g. dropdown menu), the result of the reconciliation is frequency based. The reconciled transcription will include the most commonly selected response across all volunteers. In the event that there are only two volunteers and both select a different option, the field will be omitted from the reconciled transcription. In the event that there are four or more transcribers, and half of the group chooses one option, while the other half chooses another option, then the resulting value for the reconciled field is selected between groups at random.

The process for free-text field reconciliation among multiple differing volunteer provided transcriptions is more intensive. As is detailed in the aforementioned documentation (https://github.com/juliema/label_reconciliations/blob/master/README.md), the first step is again frequency based. The most prevalent verbatim-matching entries will be chosen in the event that two verbatim matches exist. If no two field's transcriptions are identical, then a normalization step is utilized. During this step the case of the text in the field is changed to lower case, punctuation is removed, and white space characters are omitted. Then the frequency based approach is utilized again on the normalized transcriptions. If this normalization step fails to produce matching transcriptions, fuzzy string matching is utilized.

In this case, partial ratio matching is first applied, in which parts of one transcript are attempted to be identified as constituent parts of another. This is intended to address differences caused by abbreviation, such that 'rd' and 'road' are identified to be the same.

The transcript pairs with the highest matching score is isolated, and one of the two highest scoring transcripts is selected at random. In the event that partial ratio matching of the normalized text produces no two identical transcriptions, then token set ratio matching is attempted. During this process, the verbatim text of one transcript is viewed as a subset of another. For instance the paired text: 'dr' and 'drive' would result in a token set ratio match. Meanwhile, the text: 'rd' and 'road' would not result in a token set ratio match because the characters 'rd' do not appear together sequentially in the text: 'road'. In the event that the token set ratio approach fails to identify two similar transcripts, the transcriptions are deemed irreconcilable.

For the purposes of this research, the target class label 'scientificName' was produced as a result of the reconciliation process described above. It was hoped that despite the noise present in the reconciliation process, it would still be feasible to classify samples under this class label, as demonstrated by Carranza-Rojas et al. [6].

Chapter 4

Methods

Within this chapter the experiments that were conceived in accordance with the thesis goals (section 1.3) are first introduced. The experimental design section 4.1 describes the transition from the originally stated thesis goals to an experimental framework compatible with the scientific method. The experimental procedure section 4.2 describes the required functionality of the preconceived experiments in greater detail.

4.1 Experimental Design

4.1.1 Exploration of a Candidate Hyperparameter Space

As mentioned in subsection 1.3.1, it was a goal of this research to determine if there existed an optimal set of hyperparameters that could perform well across both of the datasets mentioned in chapter 3. It was deemed insufficient to arbitrarily choose a single set of hyperparameters from the conceivably infinite space of all possible hyperparameter combinations. A network trained on only a single hyperparameter sample taken from the

vast population of hyperparameter grid choices would produce results insufficient to come to a generalizable conclusion. Yet inadequate processing power exists to try all possible combinations of hyperparameter grids drawn from this space. As a result, heuristics must be used, when admissible, to narrow the search space of possible hyperparameters. For instance, it has been shown that in general, several modern optimization techniques (such as Adam and Nesterov) are more efficient for use with gradient descent than others [26].

The primary purpose of this experiment was to utilize such heuristics to determine a set of hyperparameters that performed optimally when used to train classifiers on each dataset. This was to be accomplished with the goal of isolating a particular set of hyperparameters for use in the following experiments. In the idealistic scenario in which equivalence among the optimal hyperparameter sets existed, it would have been easier to contrive of a control group for following experiments involving transfer learning. In the event that classifiers trained identically on each data set were found to have developed separate optimal hyperparameter sets, then two distinct hyperparameter sets were to be utilized as a fixed setting in following experiments.

4.1.2 Examining Feasibility of Classification

As is mentioned in subsection 1.3.1 it was a goal of this research to verify findings that suggested advanced preprocessing techniques were irrelevant for the chosen application of herbaria specimen identification. Hence it was a necessity to not only identify the one or two optimal hyperparameter sets from the previous experiment, but to compare and

assess the performance of classifiers trained using said hyperparameter sets. It was hoped that a top-1 and top-5 classification accuracy would be observed on the GoingDeeper validation dataset which was similar to that which was obtained by Carranza-Rojas et al. on their Herbaria1K dataset [6]. If a similar accuracy was obtainable on the researcher-provided GoingDeeper dataset, then there would be strong supporting evidence regarding the feasibility of classification in this domain. Importantly, such evidence would be present despite the notable absence of advanced preprocessing techniques (such as auto-segmentation), and irrespective of the presence of noisy class labels.

4.2 Experimental Procedure

The preceding section 4.1 sought to briefly introduce the conceived experiments in accordance with a high level overview. In contrast, this section seeks to describe the functionality required for the implementation of the proposed experiments to such a degree that the process is replicable. Hence, the discussion in this section is intended to be both software and framework agnostic.

4.2.1 Exploration of an Optimal Hyperparameter Space

This experiment was conceived with the intent of determining optimal hyperparameter sets to enable a fair comparison in future experiments. As such, the first stage in this experiment involves executing an identical grid search on both datasets. Due to the massive and conceivably infinite search space of all possible hyperparameter combinations, the hyperparameter-grid search space is considerably narrowed via recommendations from

recent machine learning literature.

One such heuristic is the narrowing of the optimization method utilized to be either Nesterov, or adaptive moment estimation (Adam) based optimization. In the book 'Hands-On Machine Learning with Scikit-Learn & TensorFlow', author A. Géron recommends, in general, the usage of Nesterov Accelerated Gradient during the training of deep neural networks [27]. However, multiple publications have shown that Adam based optimization compares similarly to other existing optimization methods, and can even outperform Nesterov momentum-based gradient descent [28, 26]. As a result of this, both Nesterov and Adam based optimization were to be part of the experimental grid search, whereas other more rudimentary optimization techniques such as stand-alone momentum-based stochastic gradient descent were to be ignored, admittedly primarily in the interests of computational feasibility.

Likewise, similar logic was applied to the choice in activation function as to limit an otherwise massive (albeit discrete) combinatorial search space. While originally popular due to biological prevalence, sigmoidal activation functions have largely been replaced by more modern activation functions, and can be excluded from comparison. The primary reason for their removal in large and deep neural networks is the saturation of gradients as mentioned by X. Glorot and Y. Bengio in their acclaimed paper introducing Xavier/Glorot initialization [29]. As a result of the exploding gradient problem, sigmoidal activation functions were excluded from the experimental grid search, as it had been shown that more modern activation functions perform better in such networks. Activation functions such as ReLU and ELU tended to avoid a derivative of zero at the positive and negative extremes, which prevented the dying out of neurons during the

training process. As a result, both ReLU and ELU were good candidates pertaining to a choice in activation functions for the experimental grid search.

However, the success of the chosen activation function (in terms of avoiding gradient saturation) is largely dependent upon the weight initialization scheme utilized during training [29, 26, 28]. It was previously a common practice to initialize weight values randomly in accordance with the normal distribution. However, recent research has led to the wide-scale adoption of more advanced weight initialization techniques.

In the publication proposing (the now widely used) Xavier/Glorot initialization, it is found that random weight initialization (when combined with sigmoidal activation functions) perform poorly in deep neural networks, and that this is due primarily to gradient saturation [29]. Hence, in order to benefit from modern network weight initialization techniques, the activation function must be paired appropriately. Supporting evidence comes from researcher S. Kumar, who (in published work on the ReLU activation function) asserts that the now commonplace technique of Xavier/Glorot initialization is a poor choice when combined with ReLU activation methods, which have gone on to largely replace legacy sigmoidal activation functions [30]. Therefore, in the proposed experiment, multiple weight initialization techniques are still considered, largely in an effort to verify such findings.

For the purposes of computational efficiency in the following experiments, training batch size is to be considered an additional hyperparameter for inclusion in the grid search. Researchers Carranaza-Rojas *et al.* recommend a training batch size of sixteen images [6]. Whereas, researchers Ghazi *et al.* experimented with training batch sizes of twenty and sixty images respectively, before concluding that a small but sufficient

batch size should be used in the training of large deep neural networks [7]. However, with a dataset as large as modern biological datasets it is unlikely that a batch size of sixty images is sufficient enough to include a sample from each class. This could potentially delay the training process, which is operating under early stopping criterion, by requiring considerably more epochs before the weights of all classes have had a chance to be updated appropriately. As a result, it is proposed that a larger training batch size should be utilized in proportion to the number of unique classes in the training dataset.

For simplicity's sake it was decided that a training batch size would be utilized which is the nearest power of ten to the number of distinct classes in the corresponding dataset. Hence, for the BOONE dataset which consisted of 88 unique classes, the training batch size of 100 was considered by the grid search. Likewise, for the GoingDeeper dataset which contains 851 class labels, a training batch size of 1000 was added to the hyperparameter grid search. In an effort to verify previous findings, the training batch sizes utilized by Ghazi et al. [7] of twenty and sixty images respectively, are included in the hyperparameter grid search as well.

The grid search was to be run on both datasets, with the hyperparameter grid described above, and reproduced in Table 4.1. An initial simplification step was made during the transfer learning as a fixed feature extractor approach. The additional black box classifier (see Figure 1.6) is constrained to be only a single fully-connected linear layer. This decision was not made arbitrarily, as this approach employs only the absolute minimum number of computational graph augmentations which are required in order to make functional predictions in the target domain. This was done in an effort to utilize

Table 4.1: The hyperparameter grid proposed for experimentation based off of existing machine learning literature.

Proposed Hyperparameter Grid for Experimentation				
Hyperparameter:	**Values:**			
Activation:	Leaky ReLU		ELU	
Optimizer:	Nesterov		Adam	
Initializer:	He Normal	He Uniform	Truncated Normal	
Train Batch Size:	20	60	100	1000

the hyperparameter sets produced by this experiment as an experimental control group moving forward.

Explicitly stated, this experiment was intended to take as input a hyperparameter grid search space, and produce as output for each dataset: an optimal set of hyperparameters determined within the context of the provided hyperparameter grid (i.e. Table 4.1). Had time permitted, these hyperparameter sets were to remain fixed for usage in experiments investigating the long-term thesis goals presented in subsection 1.3.2.

4.2.2 Examining Feasibility of Classification

As was previously mentioned in section 1.3, the goal of this experiment was to establish an initial performance benchmark. This was originally intended to be accomplished by replicating the experimental setup of Carranza-Rojas et al.. with the intent of observing a similar performance on the researcher-provided dataset, as was reported on in 'Going Deeper in the Automated Identification of Herbarium Specimens' [6]. However, when it became clear that the researcher-provided dataset did not match the reported upon dataset (as discussed in section 3.1.2) this constraint was necessarily relaxed. Since it was difficult to replicate the exact experimental setup of Carranza-Rojas et al.. [6], all

efforts were made to create as similar of an experimental setup as was feasible.

Two grid searches were run, one for each of the datasets described in chapter 3. The process of early stopping was employed as an optimization technique, in accordance with recommendations provided by machine learning authors: A. Géron, and S. Raschka [27, 31]. Instead of invoking early stopping immediately upon the decline of the loss metric (when measured on the validation dataset), a patience of five evaluation epochs was given. This tolerance was provided in an effort to allow the selected optimization algorithm a chance to escape local minima in the gradient space. For the purposes of the conducted research, an evaluation epoch was defined to be an epoch in which validation metrics were computed upon the entire validation dataset. For both datasets, an evaluation epoch was set to occur at the frequency of once every ten training epochs.

Instead of utilizing a modified version of the InceptionV3 classifier augmented with batch normalization [6], the default InceptionV3 classifier [32] was employed and trained on both datasets, with every hyperparameter combination detailed in Table 4.1. The holdout method was utilized to prevent overfitting, with 80% of the data retained for the training set, and 20% of the data utilized for the validation and testing datasets. The loss metric used for both grid searches was cross entropy loss, and the optimization technique employed was batch gradient descent. The model declared as the winner of each grid search was the model which obtained the highest top-1 accuracy (when computed on the validation dataset) during training. This comparison was performed at the point of training termination, either necessitated by early stopping or by the maximum number of training iterations (specified as 100,000 epochs for reasons of computational feasibility).

Chapter 5

Results

5.1 Examining Feasibility of Classification

5.1.1 BOONE Dataset Results

The winning combination of hyperparameters for the BOONE dataset grid search is reproduced in Table 5.1. The winning model employed Adam optimization with a static learning rate of 0.001. In regards to the parameter choices for Adam optimization, the value of β_1 was set to the recommended default of 0.9 and the value of β_2 set to 0.999, with an ϵ of $1E^{-08}$ [28]. The winning model took only eleven seconds to train during fine-tuning of the fixed feature extractor with a single fully-connected layer. Training was carried out for 110 epochs until early stopping was invoked. The winning model had a top-1 accuracy of 72.48% and a top-5 accuracy of 94.18% when computed on the validation set.

Table 5.1: The hyperparameter settings associated with the winning model of the BOONE dataset grid search.

Weight Initialization Technique:	He Normal
Training Optimizer:	Adam
Activation Function:	Exponential Linear Unit (ELU)
Training Batch Size:	20 samples

BOONE Dataset Top-1 Accuracy

The winning model of the BOONE grid-search obtained 100% top-1 accuracy on 43 of 88 classes in the validation dataset, as is shown in Figure 5.1. The classes which the winning classifier of the grid search failed to classify with 100% accuracy are shown in Figure 5.2. One of the common causes for low per-class accuracy is that there are not enough of the corresponding class samples in the training set to learn from. Hence, Figure 5.4 is similar to Figure 5.2, but with the coloration of the bars altered to reflect the number of samples present in the training set for each corresponding class label. In Figure 5.4 it is evident that the more classes there are in the training set, the better the neural network performs on the corresponding class in the validation dataset. From this Figure it can be seen that the classes which correspond with the worst performing accuracy have less than half the number of samples as the best performing classes.

In addition to not having enough samples in the training set, low per-class accuracy can also be obtained by computing the per-class accuracy on too few sample images in the corresponding validation set, yielding a noisy estimate of the classifiers real accuracy. As a result, Figure 5.3 shows the number of class samples found in the validation dataset on which accuracy metrics were computed for each class. This Figure reveals that some of the worst performing classes were fairly assessed with a significant number

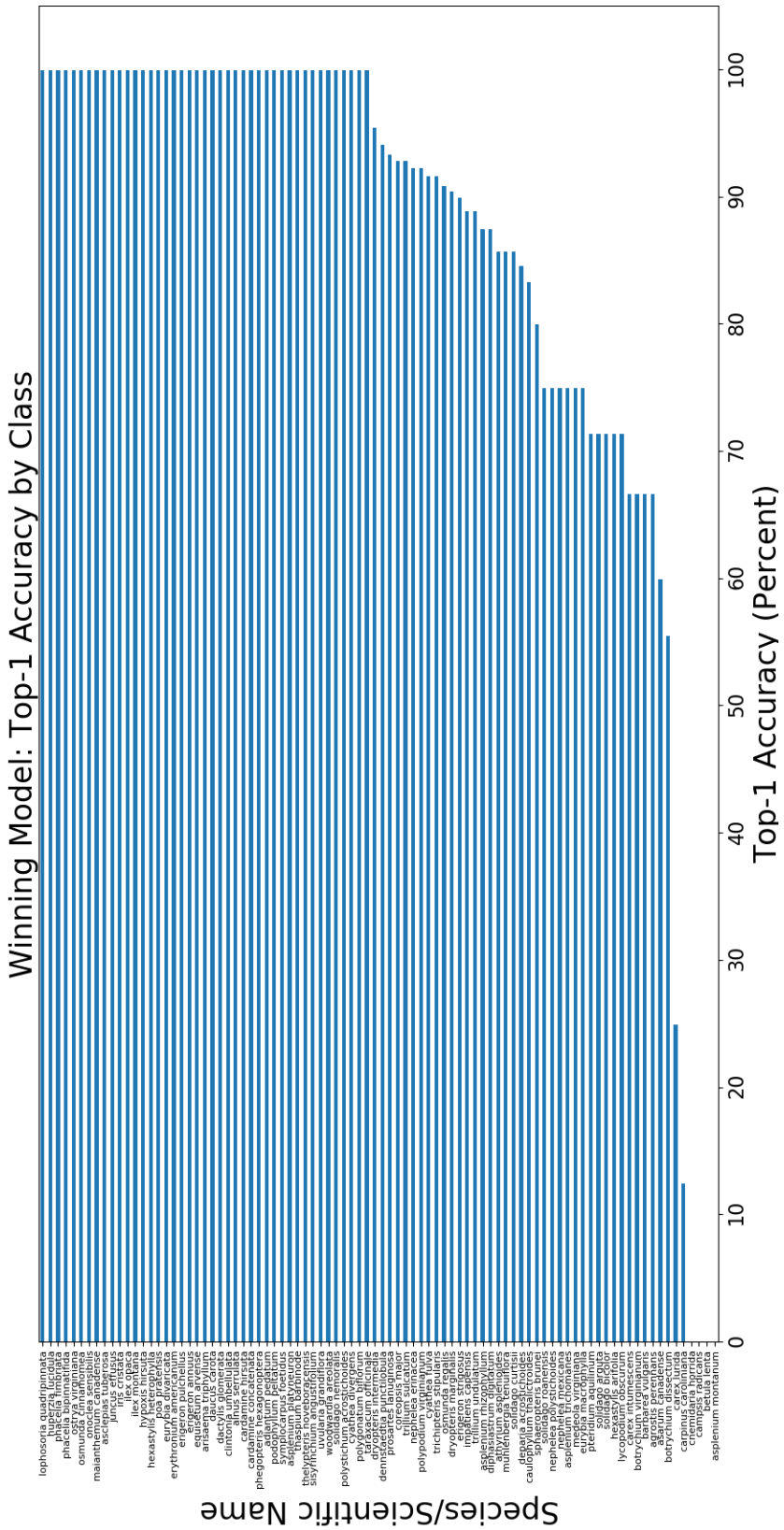

Figure 5.1: Accuracy of the winning classifier for the Boone dataset grid search, computed for each of the unique classes in the validation dataset.

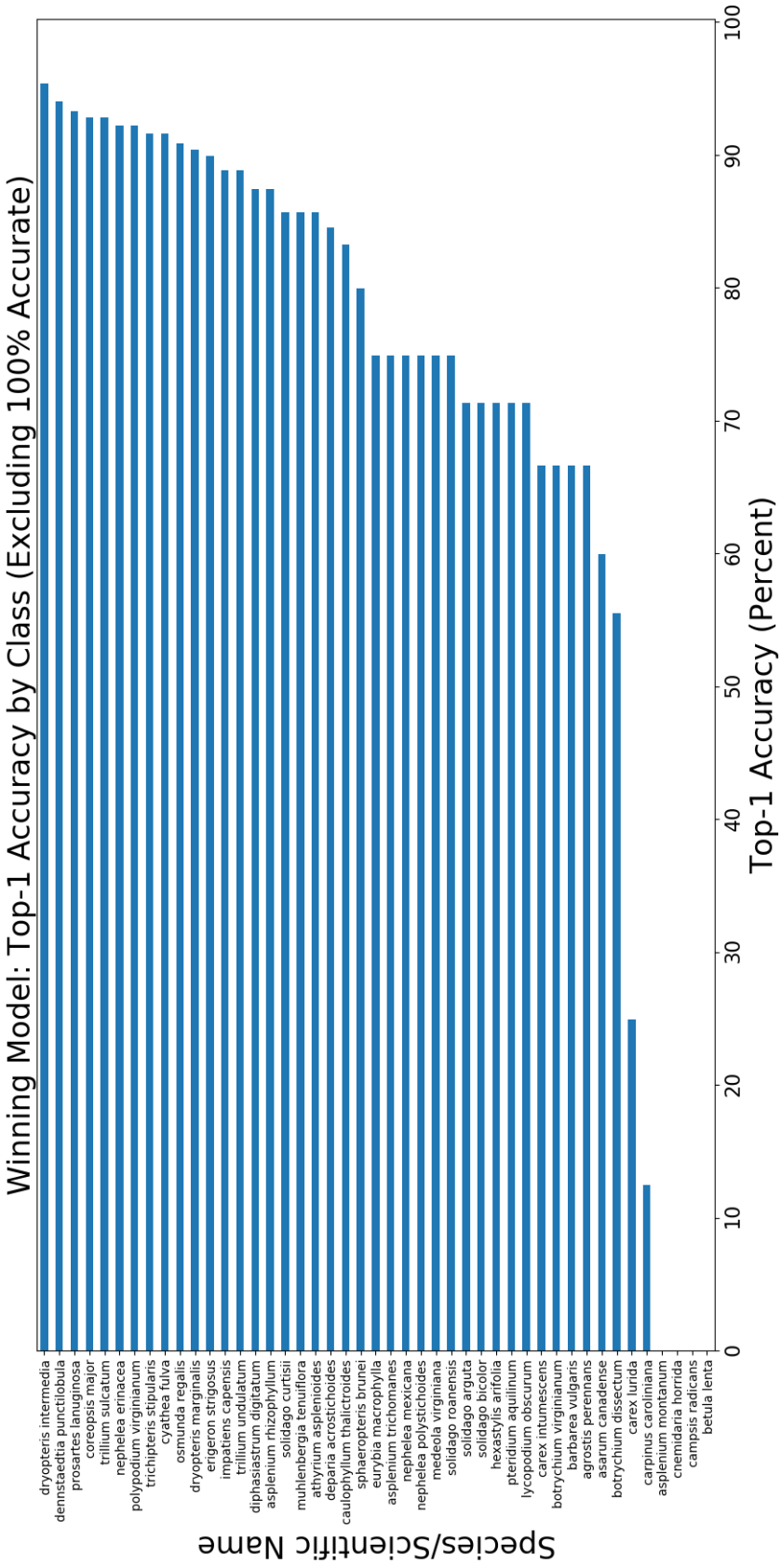

Figure 5.2: Accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset.

of validation samples. However, it also reveals that some of the worst performing classes were not evaluated adequately, as accuracy metrics were computed on less than half of the intended minimum number of samples per class.

BOONE Dataset Top-5 Accuracy

The winning model of the BOONE grid search had a correspondingly significant observed top-5 accuracy. Of the 88 unique classes in the BOONE dataset, 80 were capable of being classified with 100% top-5 accuracy (Figure 5.5). In an effort to isolate the classes where improvement is needed, Figure 5.6 shows the same view with the classes obtaining 100% top-5 accuracy having been removed. From this Figure it is clear that only eight distinct classes were incapable of being correctly identified within the top k=5 predictions 100% of the time for the validation dataset.

As was done when discussing the performance of the BOONE dataset in terms of top-1 accuracy, Figure 5.7 is identical to the preceding Figure 5.6 except for the addition of a color bar indicating the corresponding number of training images belonging to each class. The same pattern can be observed here as with the corresponding image for the top-1 accuracy performance (Figure 5.4). The more sample images that are present in the training dataset, the higher the corresponding top-5 accuracy is when computed on the validation dataset. By looking at Figure 5.8 a similar trend is observed, in that many of the classes with few samples in the training dataset were also evaluated with correspondingly few sample images in the validation set.

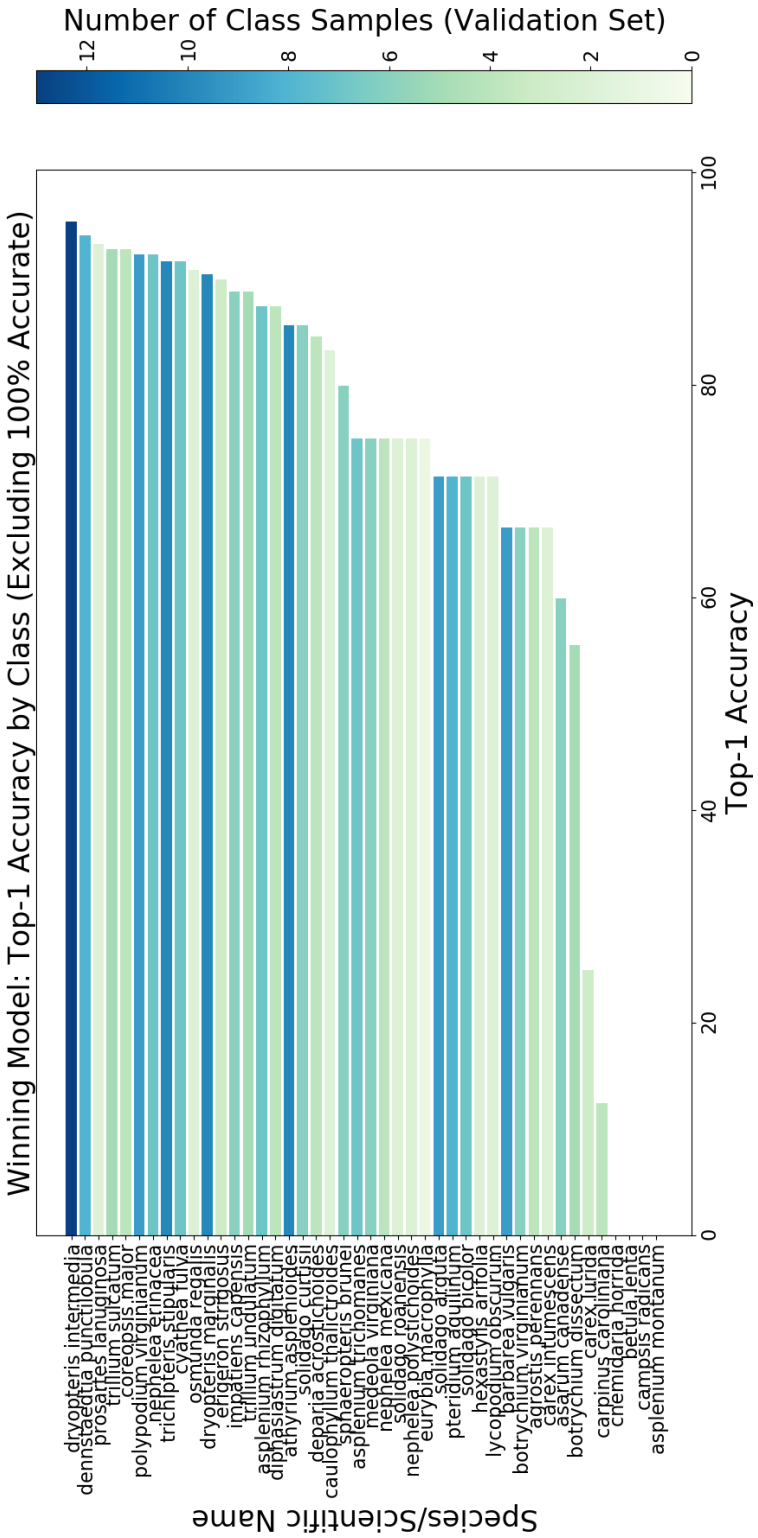

Figure 5.3: Accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the validation dataset.

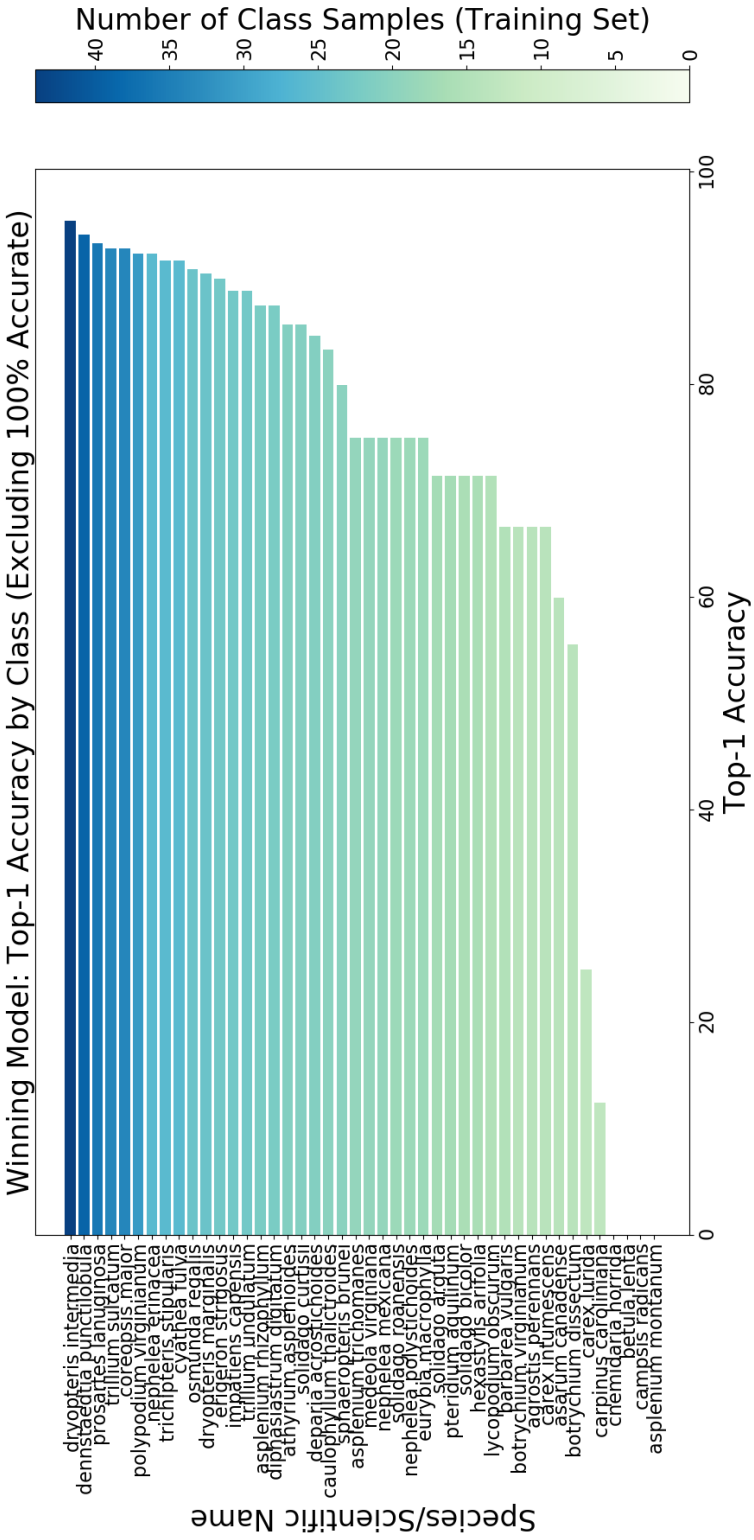

Figure 5.4: Accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the training dataset.

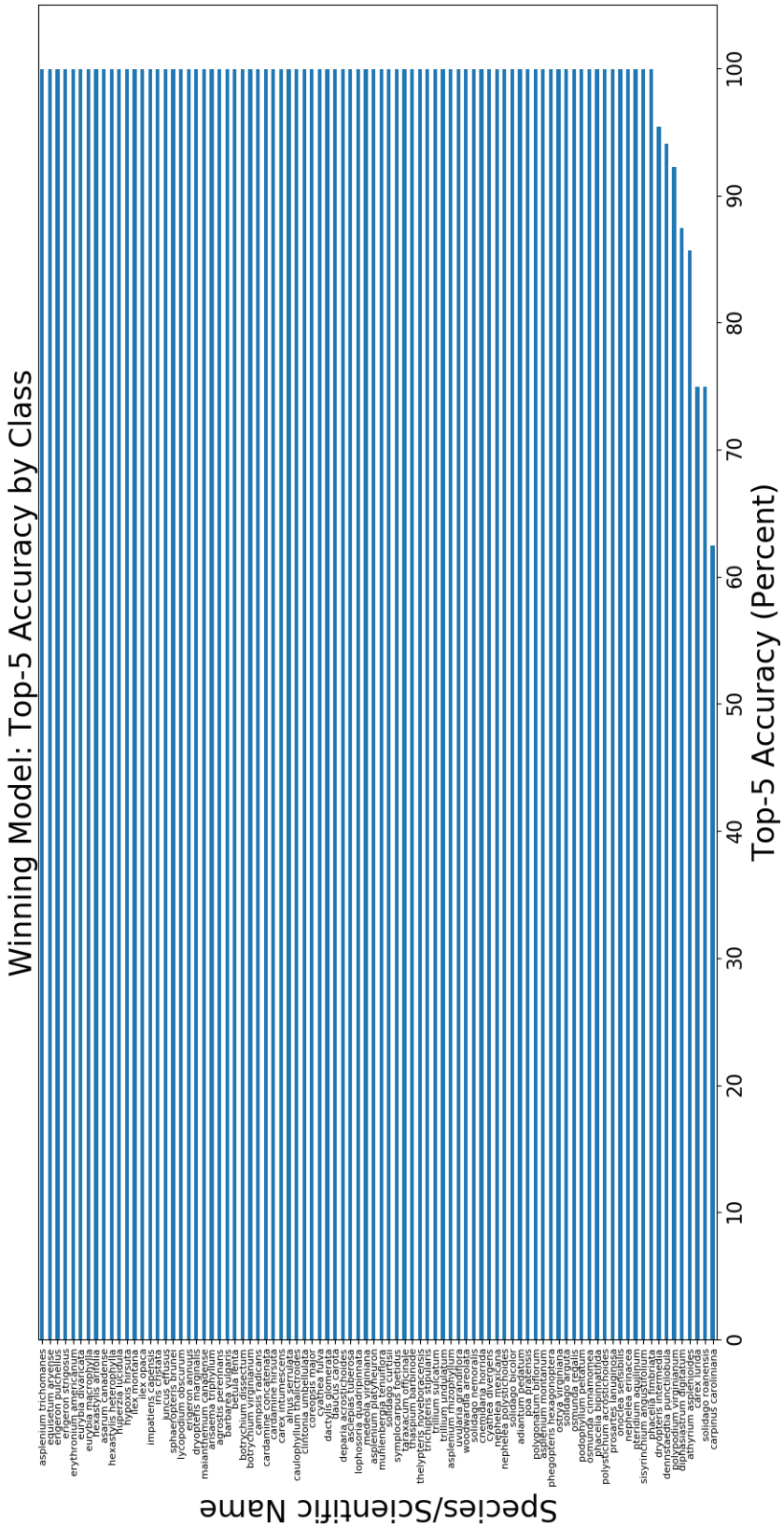

Figure 5.5: Top-5 accuracy of the winning classifier for the Boone dataset grid search, computed for each of the unique classes in the validation dataset.

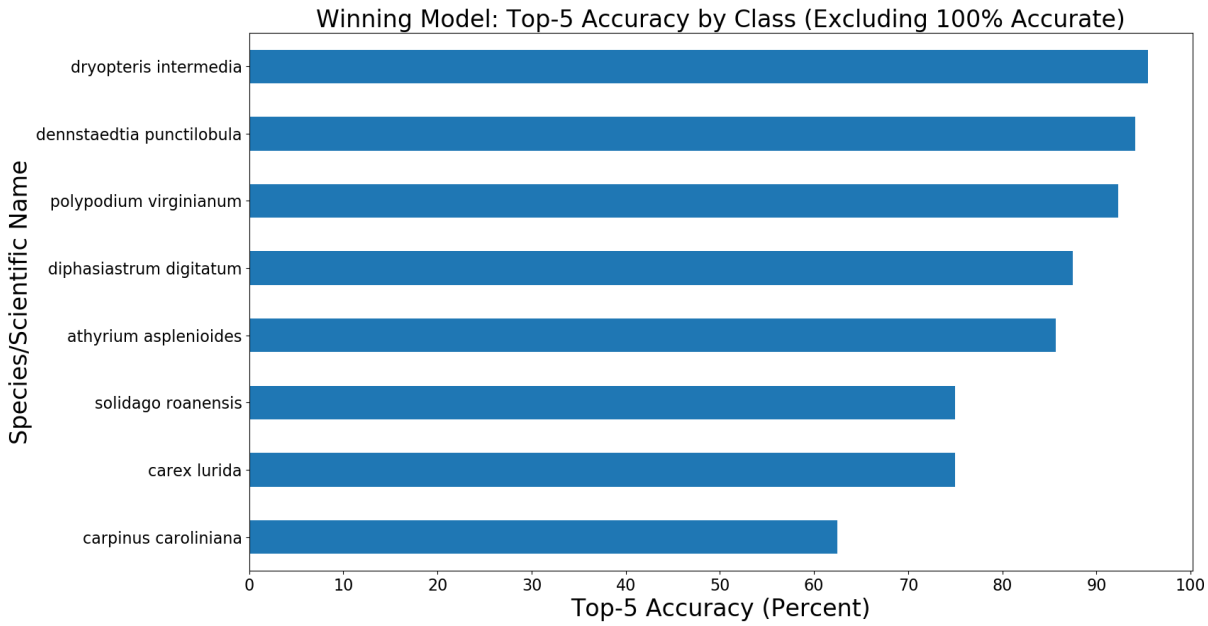

Figure 5.6: Top-5 accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset.

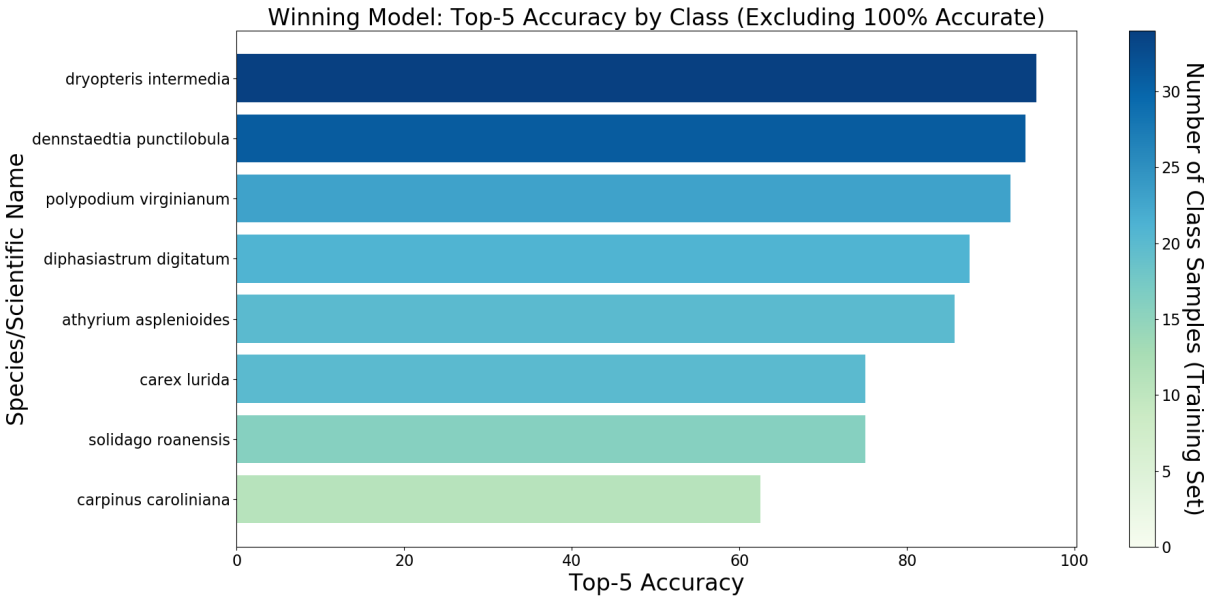

Figure 5.7: Top-5 accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the training dataset.

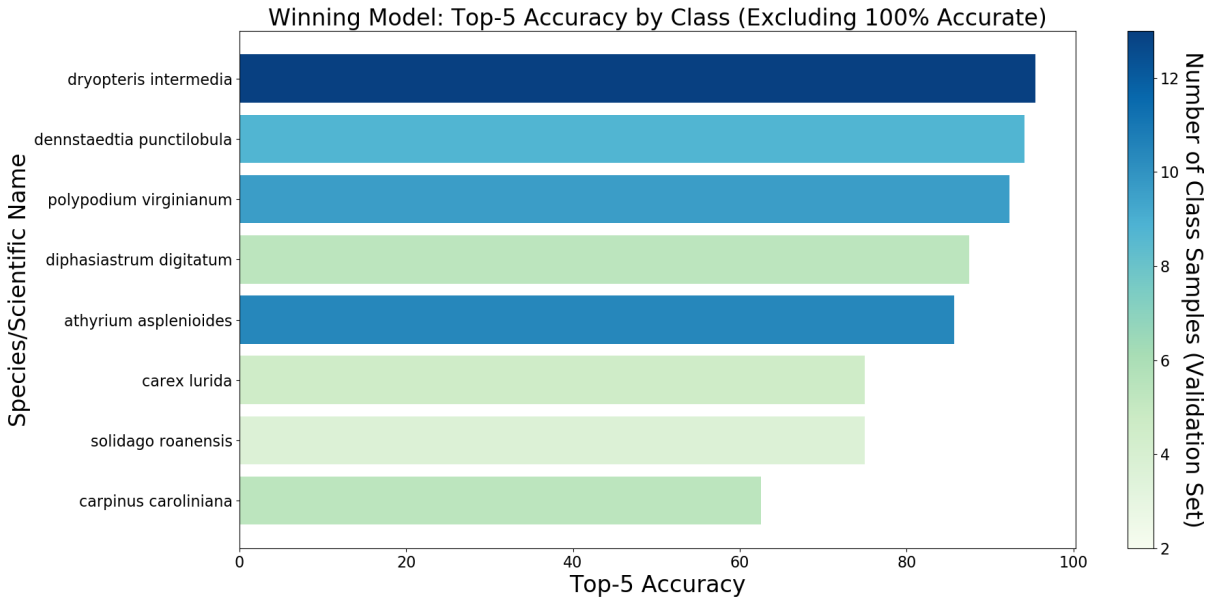

Figure 5.8: Top-5 accuracy of the winning classifier for the Boone dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the validation dataset.

5.1.2 GoingDeeper Dataset Results

The hyperparameters for the winning model of the GoingDeeper dataset grid search are shown in Table 5.2. The winning model for this dataset took 3 hours 17 minutes and 54 seconds to train, over the course of 6,115 epochs until early stopping was executed. This model obtained a top-1 accuracy of 45.36% on the validation set, and a top-5 accuracy of 68.29% on the validation set. Contrary to the results obtained by the BOONE grid search, for the GoingDeeper dataset, Nesterov optimization was observed to have performed better. An identical static learning rate of 0.001 was utilized along with a momentum constant of 0.9.

Despite possessing the largest validation accuracy, other candidate models performed almost as well on the validation dataset while requiring significantly less training time. For instance, the same identical hyperparameter settings with a training batch of

Table 5.2: The hyperparameter settings associated with the winning model of the GoingDeeper dataset grid search.

Weight Initialization Technique:	He Normal
Training Optimizer:	Nesterov
Activation Function:	Leaky Rectified Linear Unit (ReLU)
Training Batch Size:	1000 samples

twenty samples resulted in a comparable top-1 accuracy of 44.75% and a top-5 accuracy of 67.83% while requiring only 44 minutes and 37 seconds to train. This represents a significant reduction in required training time (a factor of at least one-fourth) for a net loss of only .61% top-1 accuracy and .46% top-5 accuracy. In many cases this minor loss in accuracy may be warranted given a threefold reduction in training time. Similarly, the same hyperparameter settings with a training batch of sixty samples resulted in a comparable top-1 accuracy of 45.19% and a top-5 accuracy of 68.33% while requiring only 38 minutes and 6 seconds to train.

GoingDeeper Dataset Top-1 Accuracy

The winning model of the GoingDeeper grid search failed to obtained 100% top-1 accuracy on any of the 851 unique classes in the dataset, as is shown in Figure 5.9. In relation to the BOONE dataset, possessing over nine times the number of unique classes indicates that this is a markedly harder classification problem. Whereas approximately 49% of the BOONE dataset can be classified according to top-1 accuracy with near 100% certainty, this is far from the case with the GoingDeeper dataset.

Similar to the Figures displayed for the BOONE dataset, the following images exclude the single class which obtained 100% accuracy on the validation dataset, in an effort to assess where the model performs poorly. However, since there are no classes

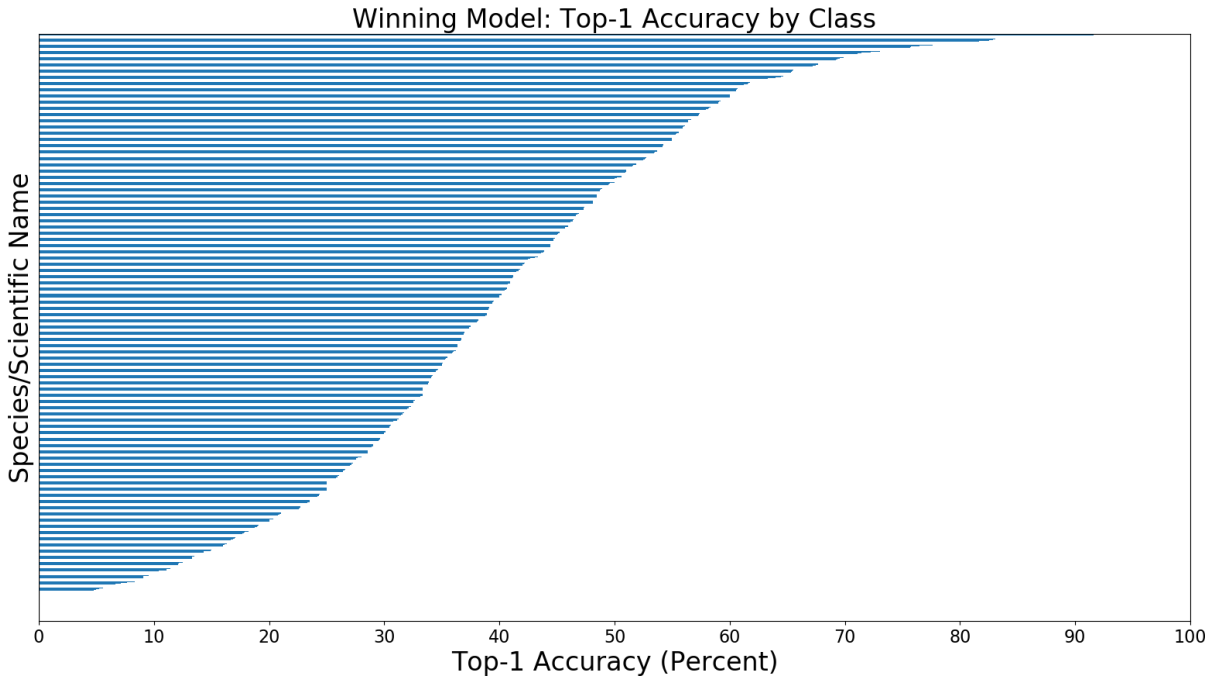

Figure 5.9: Accuracy of the winning classifier for the GoingDeeper dataset grid search, computed for each of the unique classes in the validation dataset.

with a 100% top-1 accuracy in the GoingDeeper dataset, the figures are identical. Hence, Figure 5.10 is similar to Figure 5.9, but with the coloration of the bars altered to show the number of samples present in the training set for each corresponding class label. Likewise, Figure 5.11 shows the per-class accuracy colorized according to the number of samples in the validation set. Of particular interest are classes which contain a relatively significant number of samples in the training and validation datasets, but are found to obtain a low per-class top-1 accuracy.

GoingDeeper Dataset Top-5 Accuracy

Unlike what was observed with the BOONE dataset, there was not a major improvement between the top-1 accuracy and top-5 accuracy of the winning classifier for the GoingDeeper dataset. Figure 5.12 shows the top-5 accuracy computed for each class in

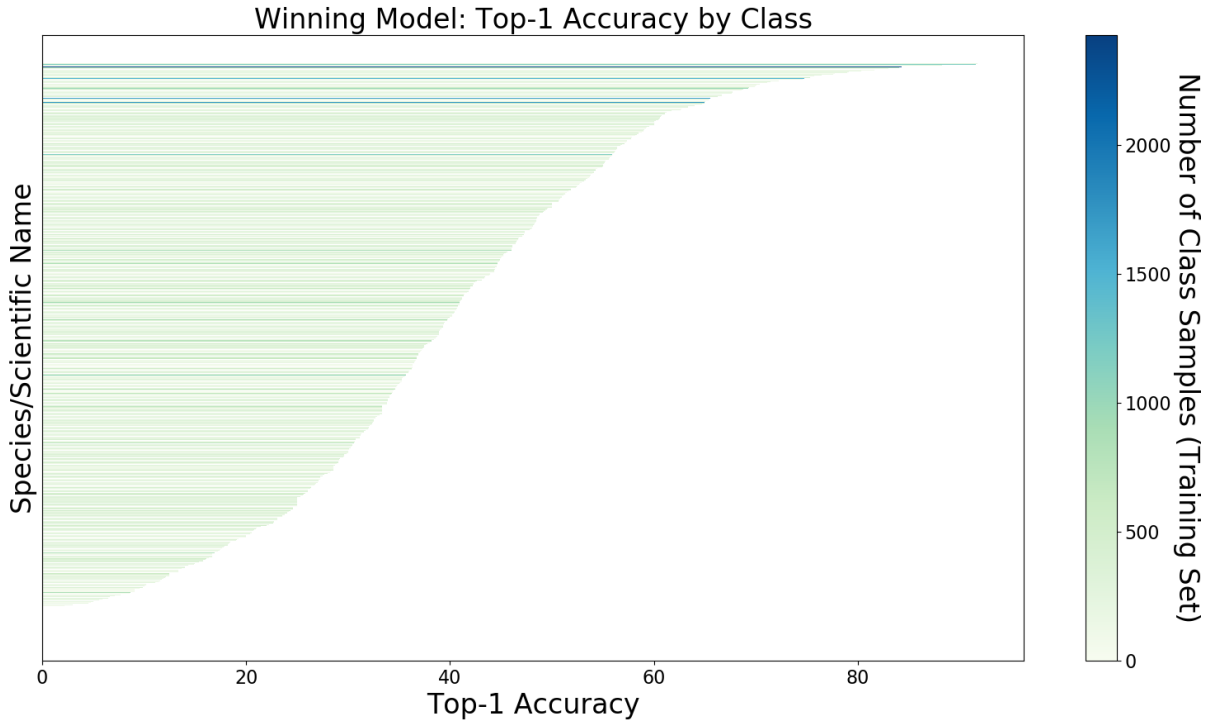

Figure 5.10: Accuracy of the winning classifier for the GoingDeeper dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the training dataset.

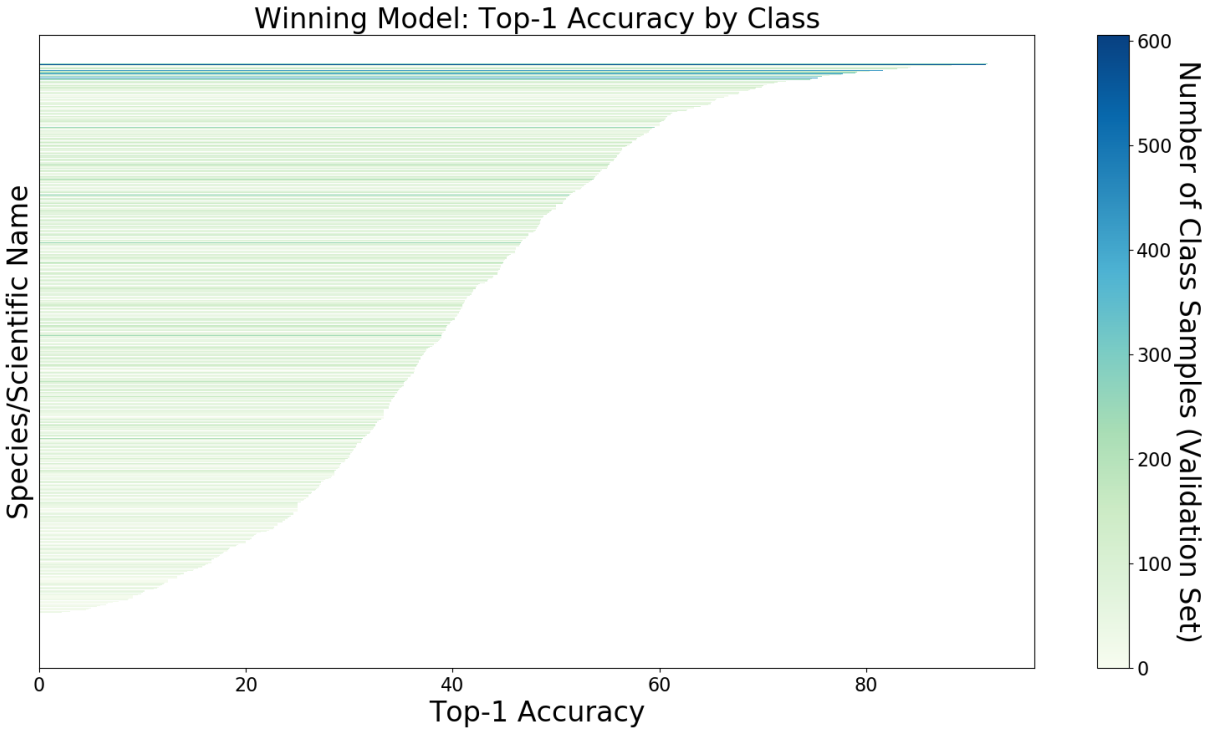

Figure 5.11: Accuracy of the winning classifier for the GoingDeeper dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class label in the validation dataset.

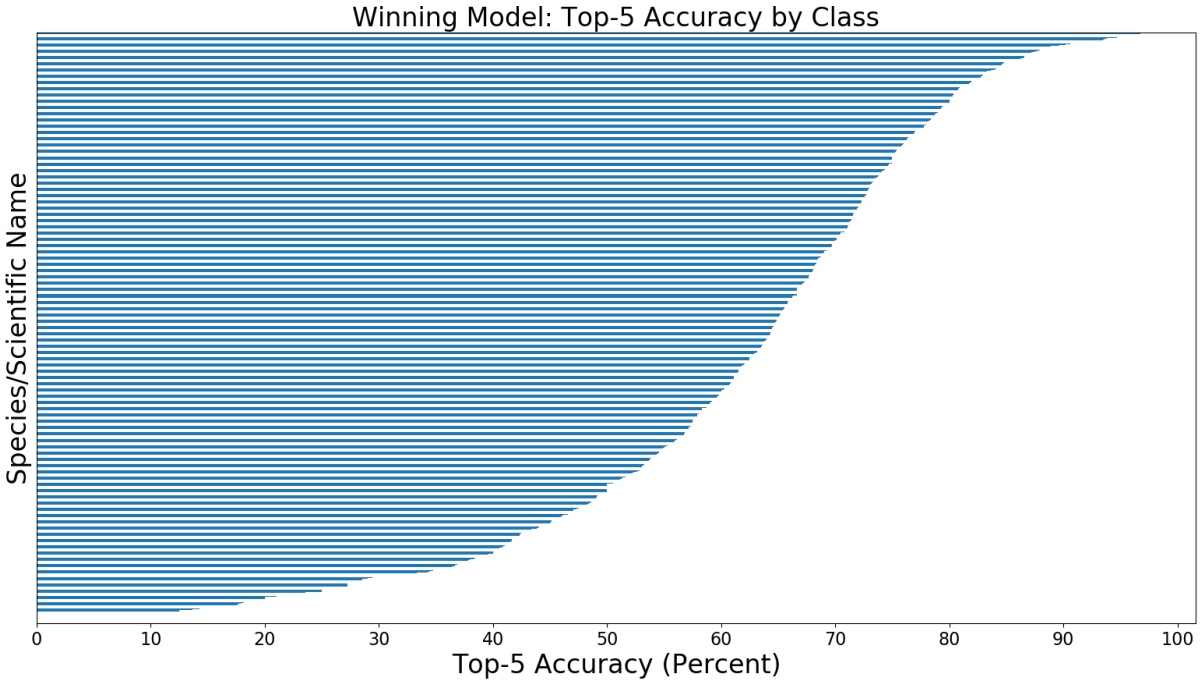

Figure 5.12: Top-5 accuracy of the winning classifier for the GoingDeeper dataset grid search, computed for each of the unique classes in the validation dataset.

the GoingDeeper dataset as measured on the validation dataset.

A similar breakdown is provided for the purposes of visualizing the top-5 performance of the classifier trained on the GoingDeeper dataset. Figure 5.13 shows the top-5 accuracy color coded in accordance with the number of corresponding samples in the training dataset. As was previously done with the results of the top-1 accuracy for each dataset, Figure 5.14 displays the top-5 accuracy for each class, color coded by the number of samples in the validation dataset by which these metrics were calculated.

5.2 Exploration of a Candidate Hyperparameter Space

There was not a single set of hyperparameters which performed the best across both datasets (see the disjoint table 5.1 and table 5.2). The method of optimization, activa-

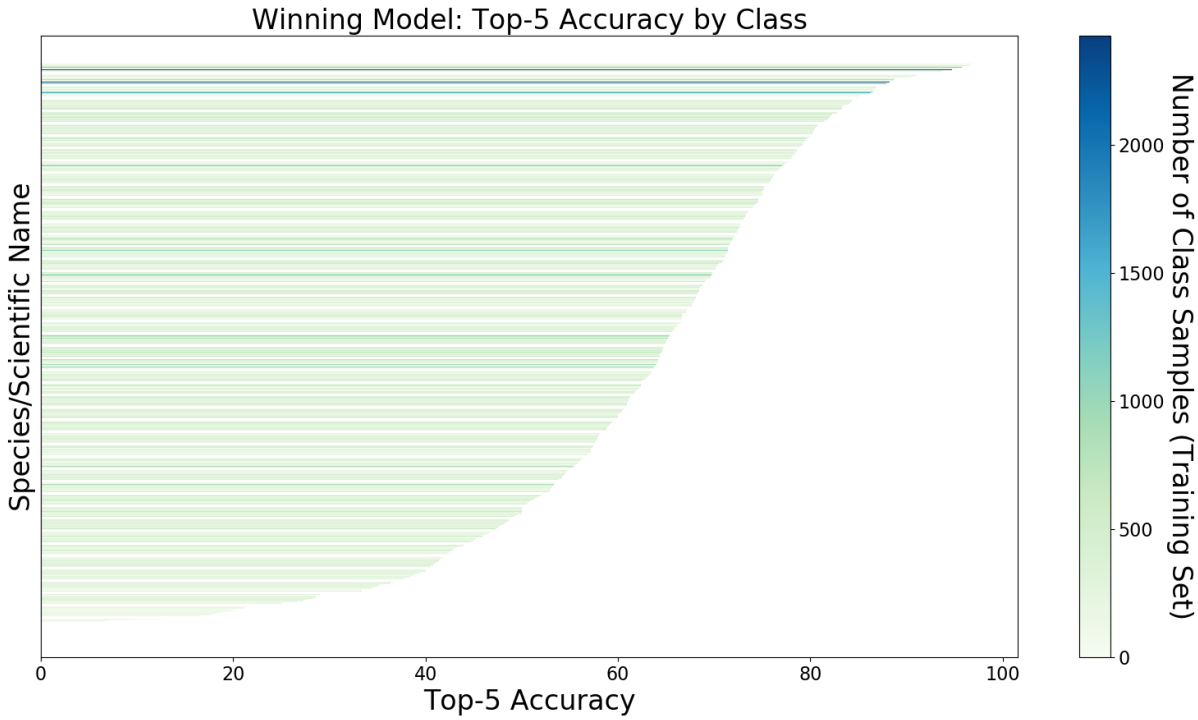

Figure 5.13: Top-5 accuracy of the winning classifier for the GoingDeeper dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class in the training dataset.

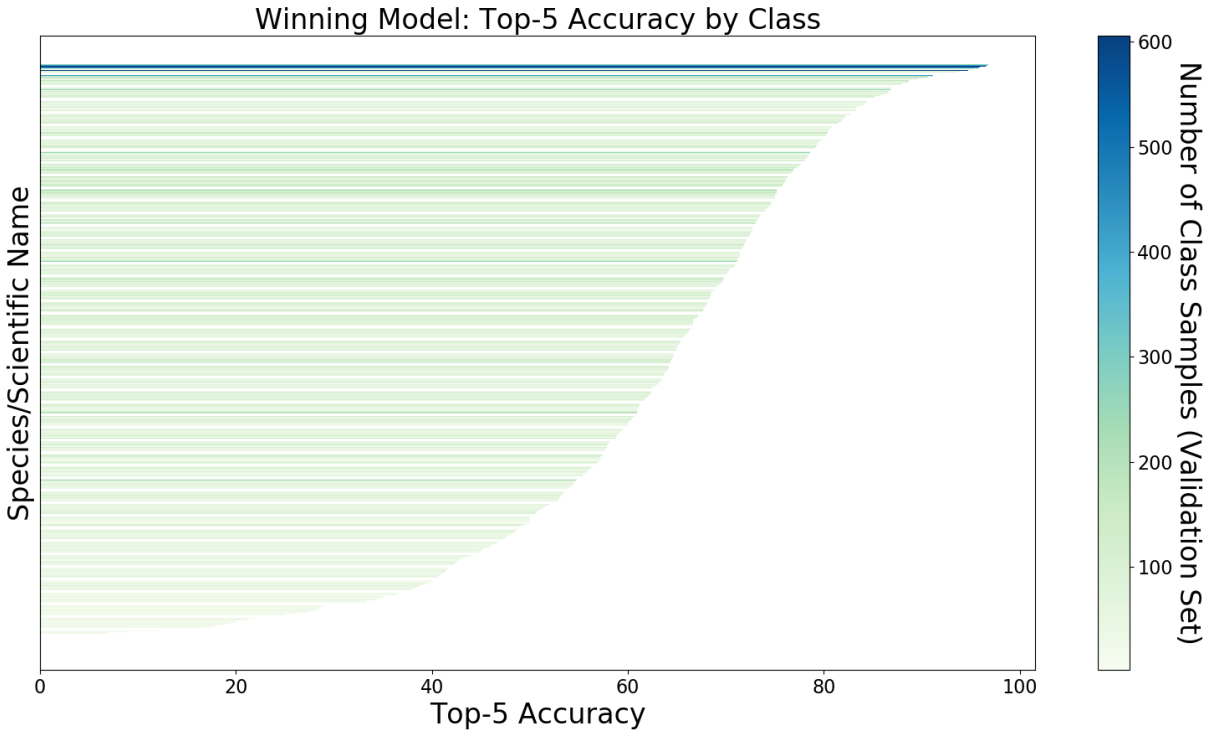

Figure 5.14: Top-5 accuracy of the winning classifier for the GoingDeeper dataset grid search (excluding classes that are 100% accurate), computed for each of the unique classes in the validation dataset. The bars are colorized according to the number of samples belonging to each respective class in the validation dataset.

tion function, and training batch size, all differed among both datasets. Although the hyperparameter sets themselves do not reveal any trends which are common among both datasets, the hyperparameter heat maps show patterns described in the following section.

5.2.1 BOONE Dataset Results

Figure 5.15 shows a heat map of the hyperparameter combinations performed during the grid search for the BOONE dataset, colorized according to the cross entropy loss metric computed during training. This Figure reveals that performing a weight initialization of the linear-based classifier in the target domain utilizing a truncated normal distribution is a poor choice. This is further evidenced by Figure 5.16, in which it is shown that the worst performing hyperparameters for the BOONE dataset (in terms of top-5 accuracy) are those in which the normal truncated distribution are utilized as a hyperparameter. As expected, this same choice in initializer translates to an even poorer top-1 accuracy, as is evidenced in Figure 5.17. This trend is equally prominent in the GoingDeeper validation set's hyperparameter heat map (Figure 5.21).

In terms of the other hyperparameters and their relation to the BOONE dataset, there are several other observations worth explicitly stating. Figure 5.18 is a box plot which highlights the role that the training batch size plays in relation to training time. For the BOONE dataset it is clear that a training batch size of 1000 specimen images leads to an exceedingly high training time of the corresponding model. From this image, it would appear that for the BOONE dataset a training batch size of 20, 60, or even 100 specimen images is sufficient. A training batch size of 1000 images results in a median

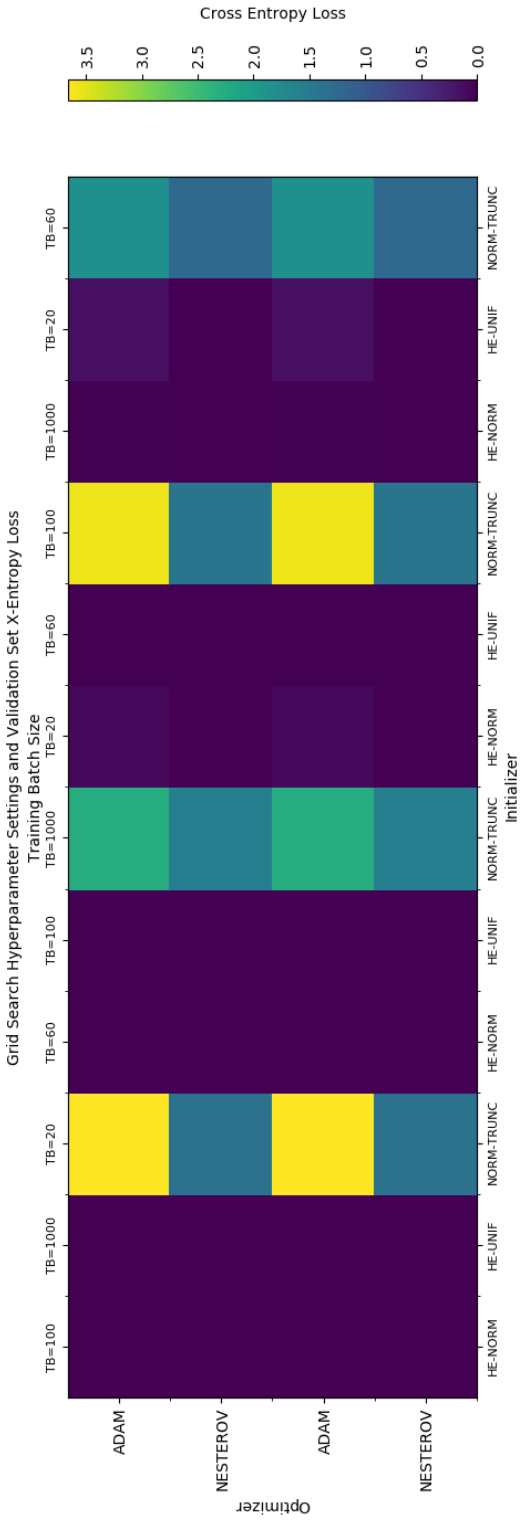

Figure 5.15: A heat map of the hyperparameters for the BOONE dataset colorized according to the cross entropy loss metric computed during training.

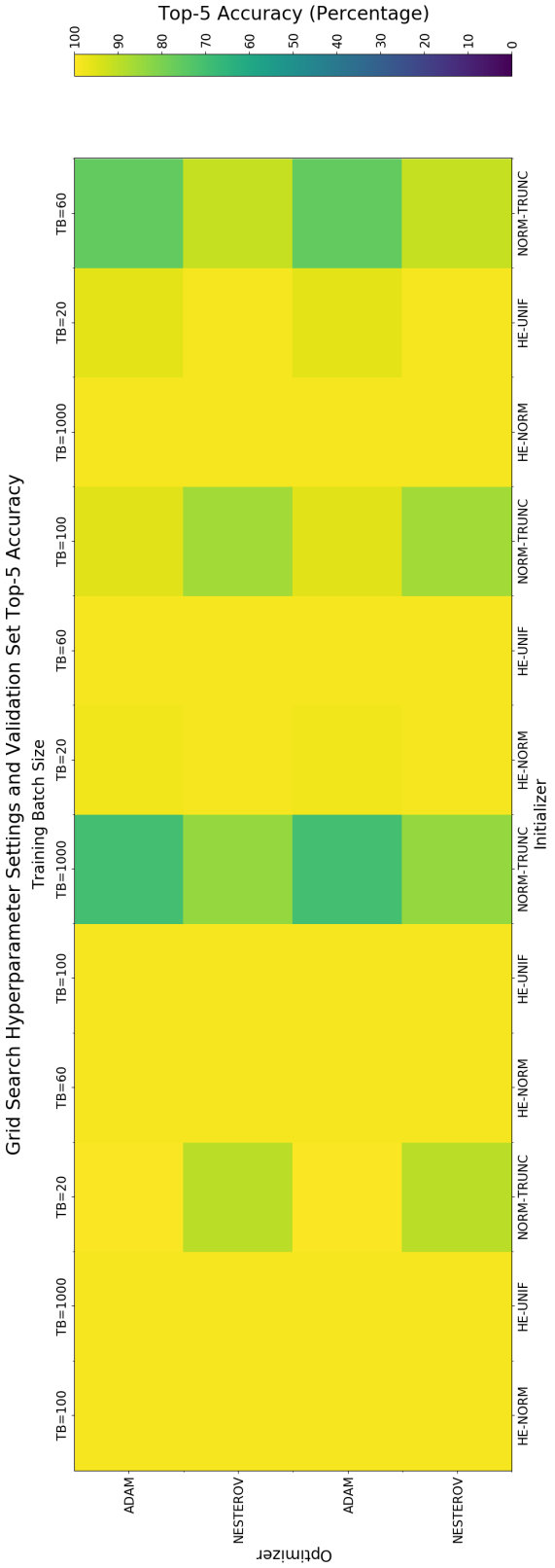

Figure 5.16: A heat map of the hyperparameters for the BOONE dataset colorized according to top-5 accuracy computed on the validation dataset.

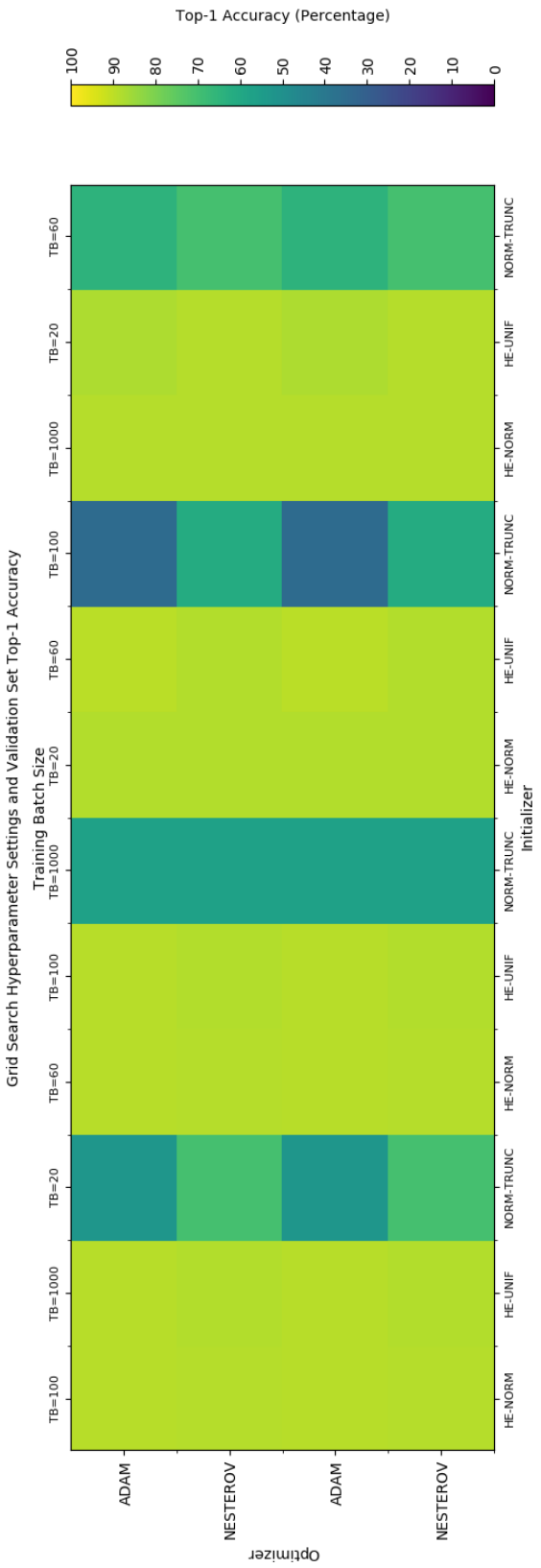

Figure 5.17: A heat map of the hyperparameters for the BOONE dataset colorized according to top-1 accuracy computed on the validation dataset.

training time that is at least twice as long as the others.

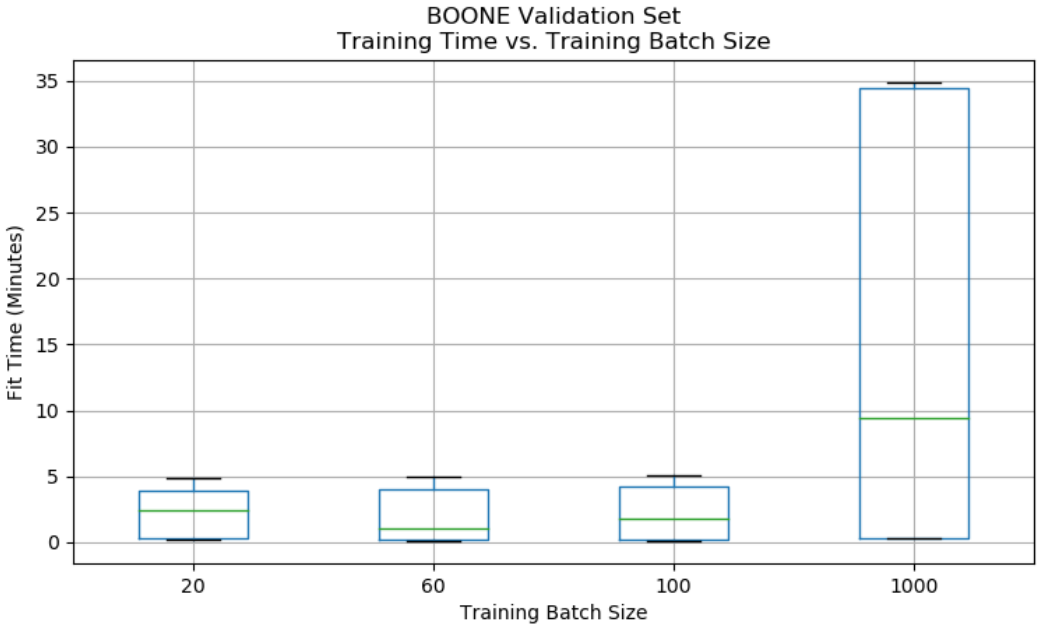

Figure 5.18: A box plot showcasing the impact of training batch size on the average number of minutes required to train a classifier for the BOONE dataset.

Figure 5.19 shows the role that the chosen activation function plays in relation to training time. From this Figure, it is evident that the choice in activation had a negligible impact on model training time. However, the same pattern was not observed when it came to a choice in the method of optimization. Figure 5.20 displays the impact that the optimization method has on training time. From this image it is evident that Adam optimization was much more efficient then Nesterov based optimization, on average, for this particular dataset. It is also evident from this Figure that both optimization methods possessed outliers which took an exceedingly long time to train in relation to other models trained via the same optimization technique. This reiterates the fact that although the choice in optimization method is significant, other hyperparameters play an important role in convergence, to the point where any potential gain in performance

sought by choosing a more expedient optimization method may be discounted in entirety.

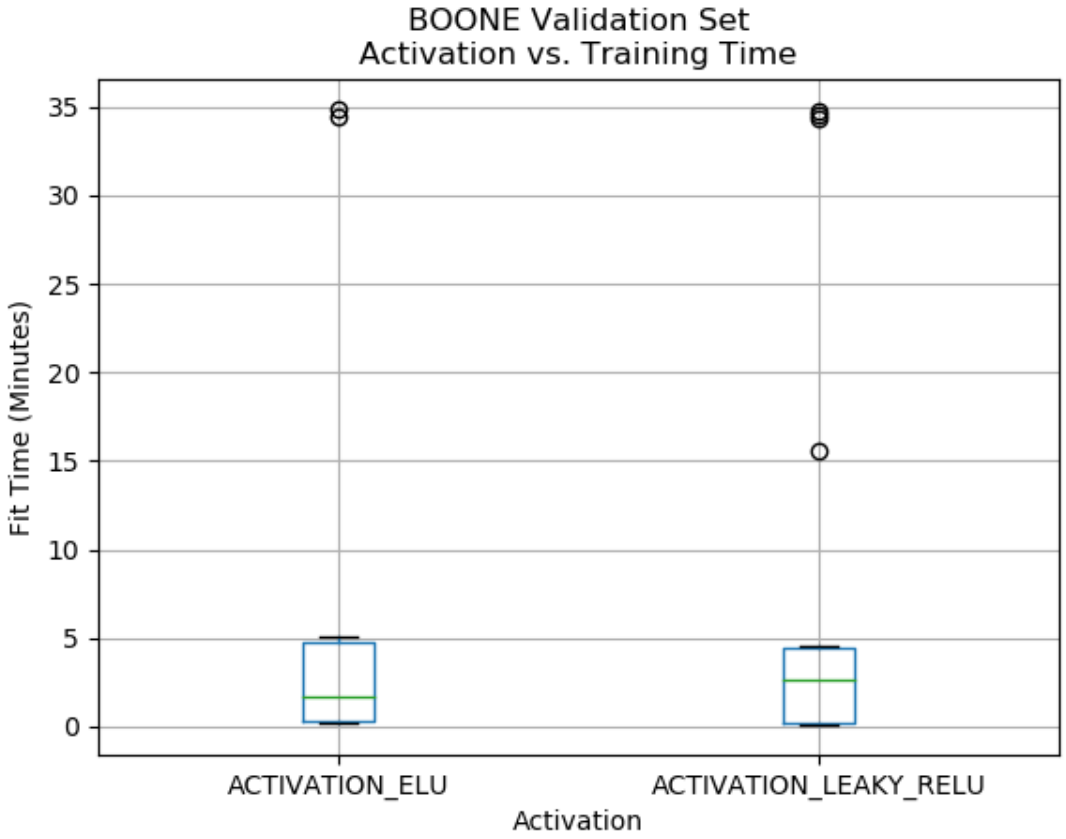

Figure 5.19: A box plot showcasing the negligible impact of the choice in activation function, on the average number of minutes required to train a classifier for the BOONE dataset.

5.2.2 GoingDeeper Dataset Results

Similar to the BOONE dataset, Figure 5.21 shows a heat map of the hyperparameter combinations performed during the grid search for the GoingDeeper dataset, colorized according to the cross entropy loss metric computed during training. Coinciding with the BOONE dataset, this Figure displays further evidence supporting the fact that performing a weight initialization via the truncated normal distribution results in poor model

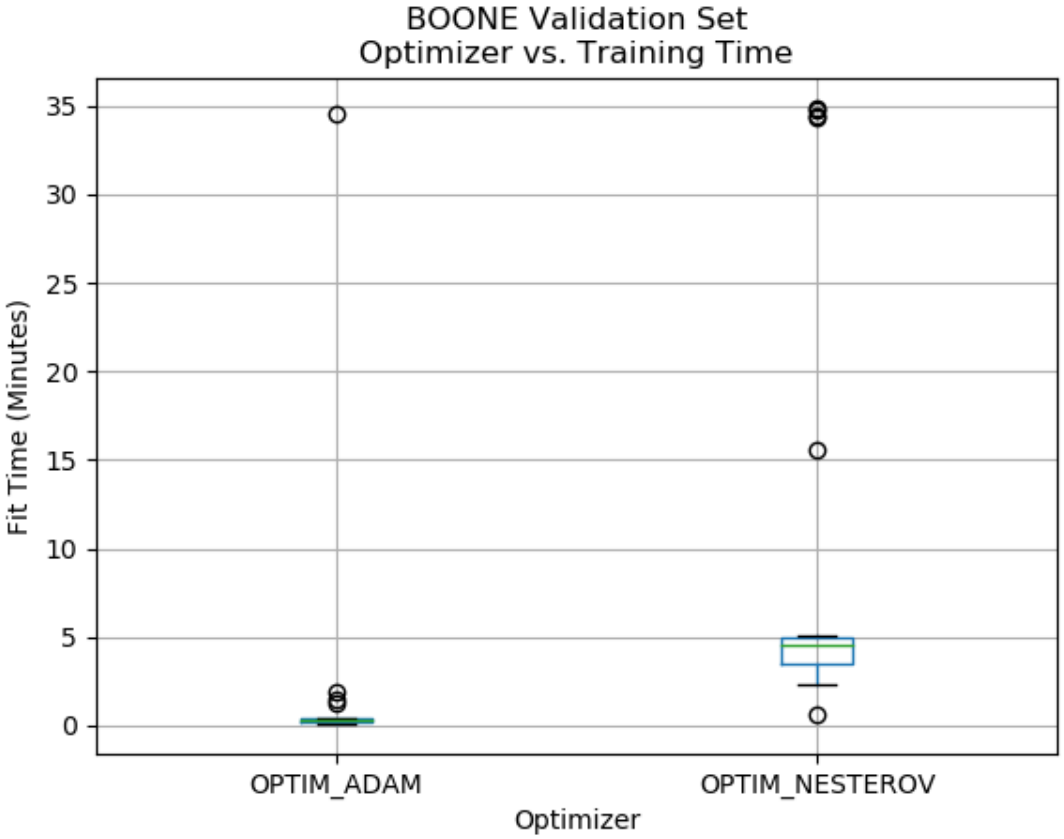

Figure 5.20: A box plot showcasing how Adam optimization is much faster than Nesterov based optimization (on average) for classifiers trained on the BOONE dataset.

performance. The corresponding heat maps for the top-5 accuracy (Figure 5.23) and top-1 accuracy (Figure 5.22) serve to further support this assertion.

Notably different from the corresponding Figure for the BOONE dataset, Figure 5.24 shows the trade-off between batch size and model training time on the GoingDeeper dataset. Unlike in the corresponding Figure 5.18 for the BOONE dataset, it is not clear that a training batch size of 1000 images is unnecessarily large for the GoingDeeper dataset, in the sense that all training batch sizes display a similar distribution of training times. Yet, there is a notable exception with a training batch size of 1000, during which at least one outlier occurred in which the maximum number of training iterations was reached.

Similar to the results observed with the BOONE dataset, Figure 5.25 asserts that the choice in activation function has a negligible impact on model training time. Both choices in activation function result in a similar distribution of training times with the exception of a consistent outlier. The same conclusion in regards to importance of choice in optimization function is found within the GoingDeeper dataset. Just like in Figure 5.20 for the BOONE dataset, Figure 5.26 for the GoingDeeper dataset asserts that Adam optimization is much more efficient than Nesterov based optimization in this domain. Unlike in the BOONE dataset however, Nesterov based optimization outperforms Adam based optimization in the winning model from the grid search for the GoingDeeper dataset. Hence, although the performance gains in terms of speed when utilizing Adam optimization are made clear from this Figure, it is important to note that the performance of the resulting classifier is not taken into account by this visual representation.

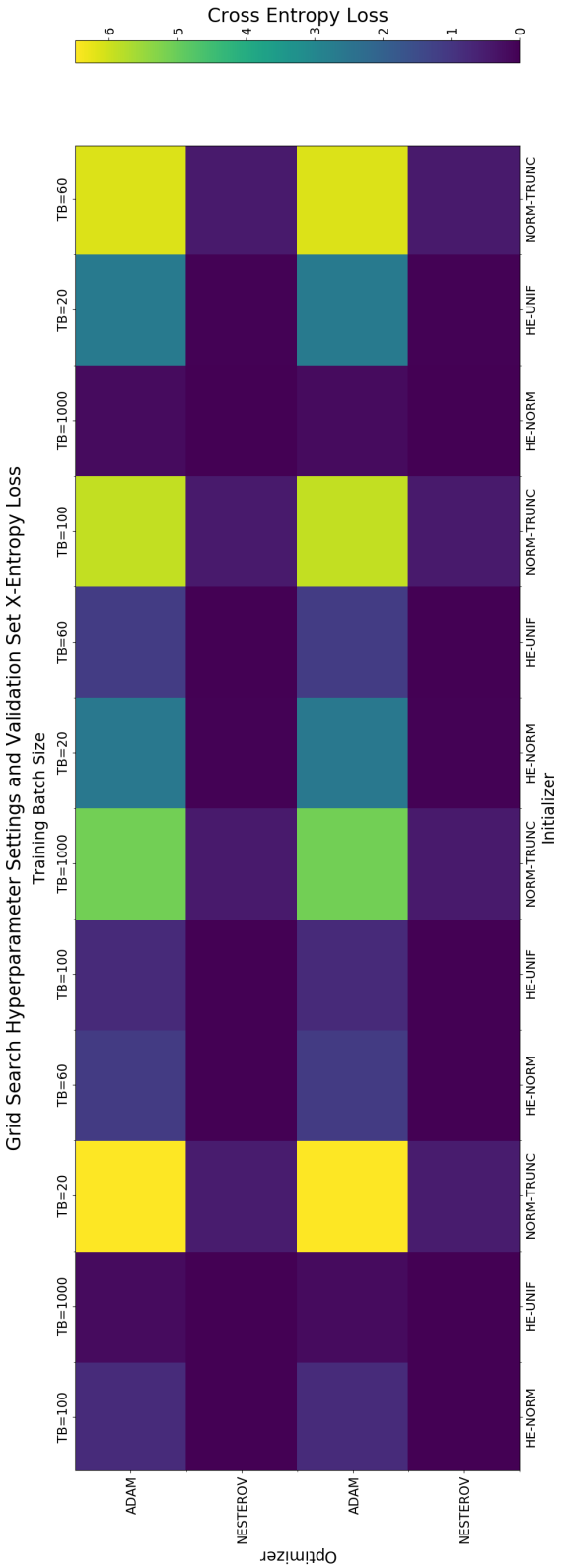

Figure 5.21: A heat map of the hyperparameters for the GoingDeeper dataset colorized according to the cross entropy loss metric computed during training.

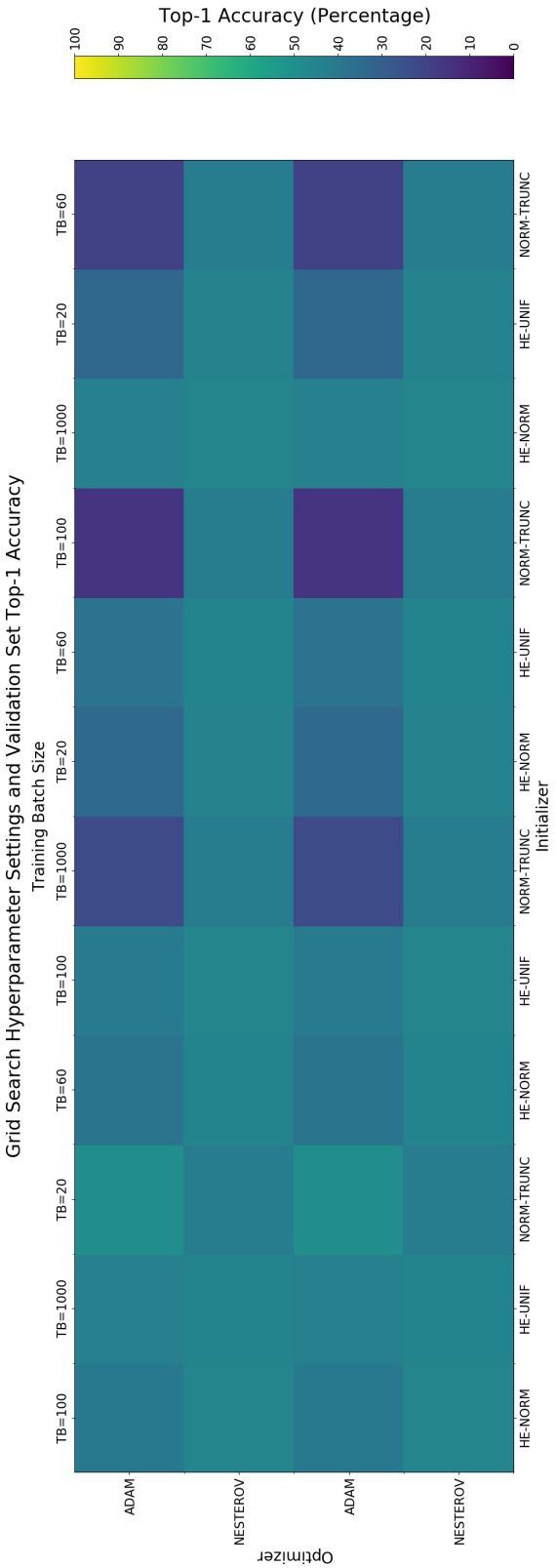

Figure 5.22: A heat map of the hyperparameters for the GoingDeeper dataset colorized according to top-1 accuracy computed on the validation dataset.

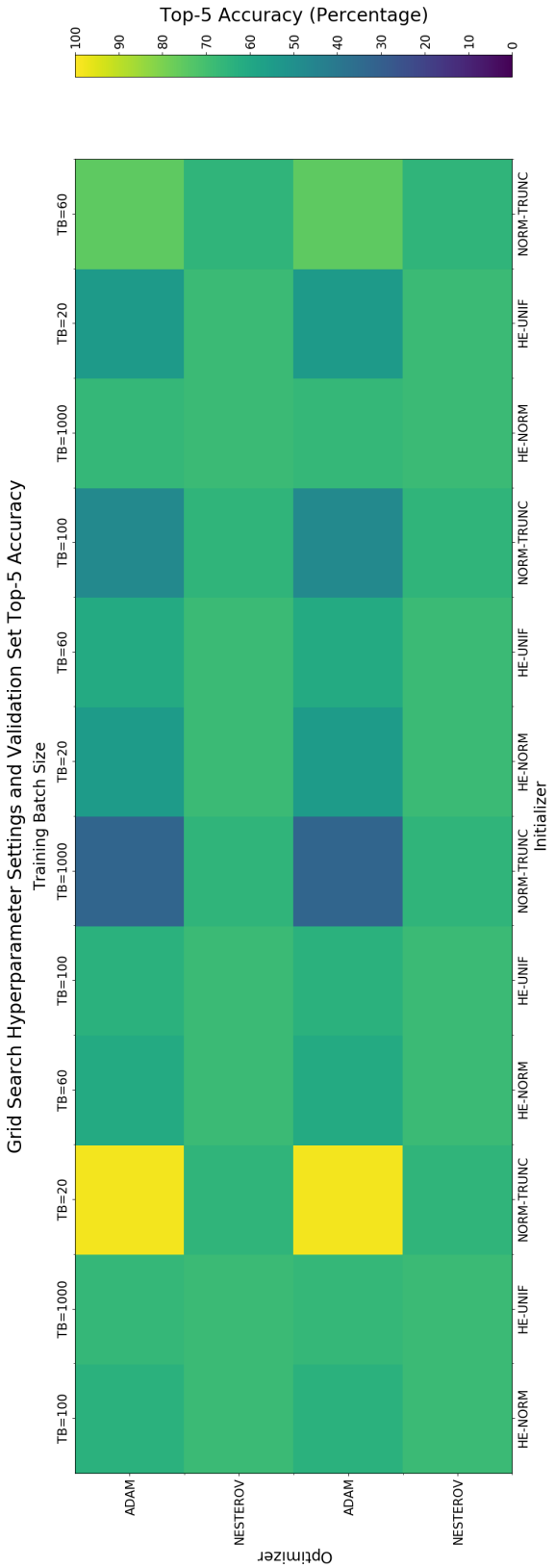

Figure 5.23: A heat map of the hyperparameters for the GoingDeeper dataset colorized according to top-5 accuracy computed on the validation dataset.

Figure 5.24: A box plot showcasing the impact of training batch size on the average number of minutes required to train a classifier for the GoingDeeper dataset.

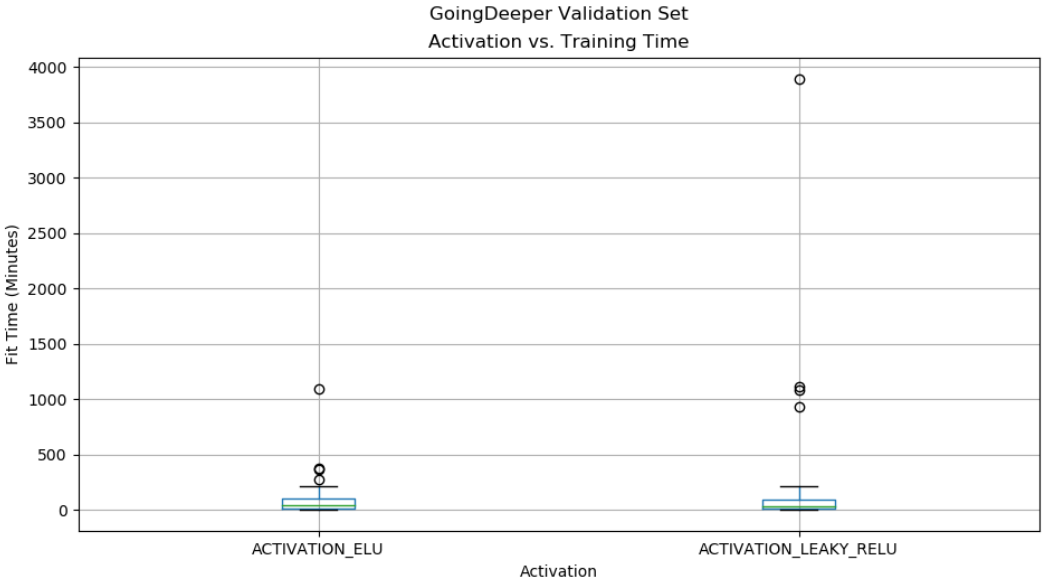

Figure 5.25: A box plot showcasing the negligible impact of the choice in activation function on the average number of minutes required to train a classifier for the GoingDeeper dataset.

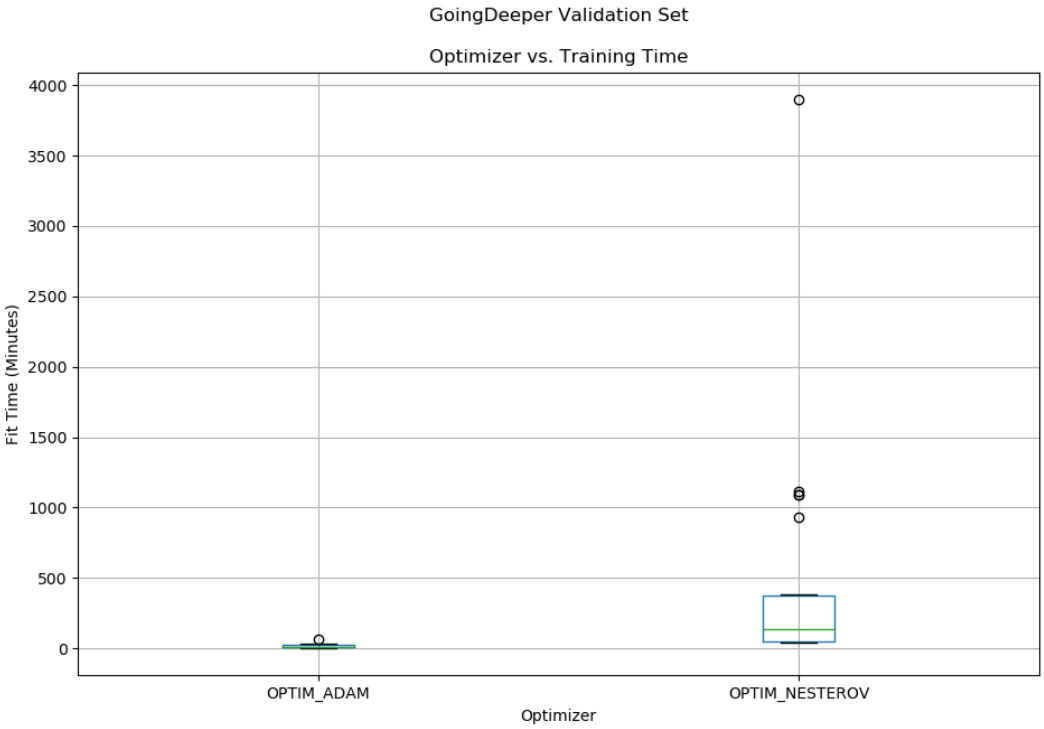

Figure 5.26: A box plot showcasing how Adam optimization is much faster than Nesterov based optimization (on average) for classifiers trained on the GoingDeeper dataset.

5.3 Discussion

5.3.1 BOONE Dataset

As was mentioned in section 5.2 it was observed for both datasets, that utilizing a truncated normal distribution for weight initialization resulted in the lowest obtained accuracy metrics. These results support the findings of previous studies which have observed that weight initialization from the normal distribution when combined with certain activation functions leads to gradient saturation which may plateau and even halt the learning process [29, 30]. Specifically, these results support the findings of S. Kumar in that non-linearities in the chosen activation function play an important role in determining a proper weight initialization scheme [30]. Additionally, these findings support the recommendations issued by existing machine learning literature which claim that He and Xavier initialization generally work well across a wide range of experimental setups, when compared to the weight initialization scheme utilizing a normal distribution [27].

Regarding the impact of training batch size on model training time, the work presented in subsection 5.1.2 (and displayed in Figure 5.18) supports the conclusion drawn by Ghazi et al. in that a small but sufficient training batch size should be preferred in the training of deep neural networks (such as that of the InceptionV3 network) [7]. From the findings presented within, it would appear that the simplification of utilizing the nearest power of ten to the number of unique class labels in the dataset, is surprisingly effective in determining a sufficient training batch size. Of course, with such a small sample size, this observation is not readily generalizable, but may merit more experimentation in the future. It would appear that selecting a training batch size that maintains a high

probability of containing a unique sample from every class may yet prove to be beneficial for mini-batch based gradient descent optimization problems.

When it comes to the selection an optimization method, there are several observations worthy of discussion. The results stated within section 5.1 assert that Adam optimization performs more quickly than Nesterov based optimization, as was shown in Figure 5.20. This was not unanticipated. Whereas Nesterov based optimization calculates the gradient with respect to the estimated future position of the parameters given the past momentum and gradient, Adam optimization computes adaptive learning rates for each parameter, performing larger gradient updates for infrequent parameters [26]. This allows Adam optimization to take advantage of sparse features and obtain comparably faster convergence rates [28]. This trend was observed in the TensorFlow training logs published digitally alongside this thesis. Of considerable interest is the fact that Nesterov based optimization outperformed Adam based optimization on the GoingDeeper dataset grid search. Why this may be is a cause for speculation which would require additional investigatory research.

Another point of particular interest is why some classes perform so well on the BOONE dataset and others perform so poorly. The worst performing class according to both the top-1 and top-5 accuracy metrics for the BOONE dataset was *Carpinus Caroliniana*. Figure 5.27 shows all sample images for this particular species. From this figure, it is easy to speculate why the classifier may be performing so poorly on this class. Of immediate concern is the variations in visual features among samples of this specimen. Whereas some images possess leaves, other sample images contain only stems with flower buds. In addition, despite image post-processing techniques designed to per-

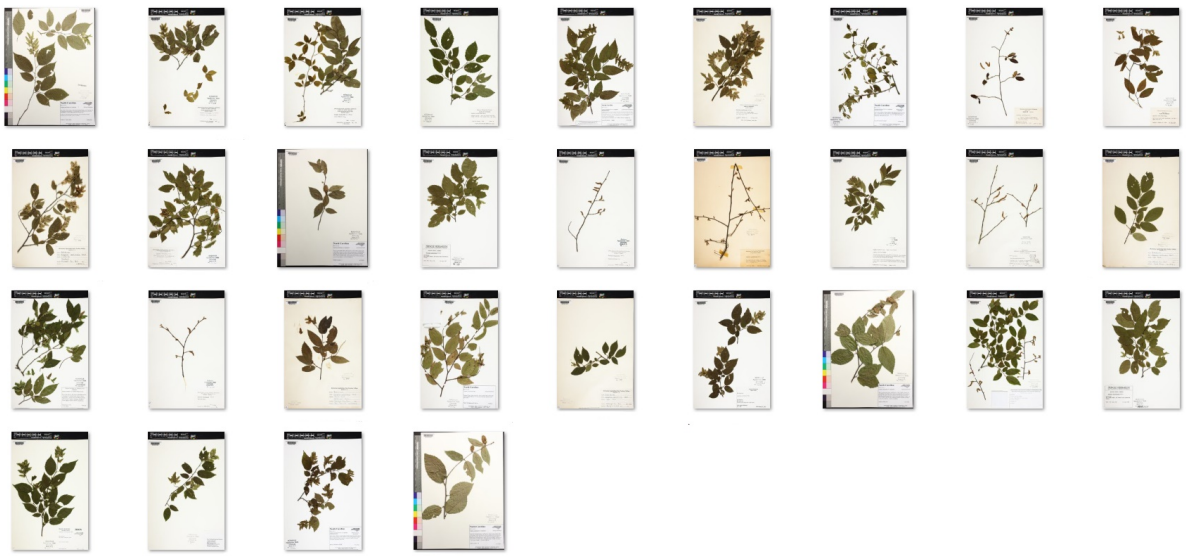

Figure 5.27: BOONE sample images belonging to the worst performing class *Carpinus Caroliniana* in the group of classes which do not possess 100% top-5 accuracy.

form color correction, the backgrounds of several of the specimen images are discolored. Although normalization is performed during feature extraction, it may be the case that the variations in background color is influencing the classifier's output. The fact that there are so few total sample images (only 31) is also an area of concern.

Meanwhile, the best performing class in the group of non-100% top-5 accurate classes, is the species *Dryopteris Intermedia*. Looking at sample images for this species (Figure 5.28) it is easy to see why the corresponding top-5 accuracy for this class may be so much higher comparatively speaking. Not only is there relatively minor morphological differences between specimen images, but the background coloration is largely consistent across all samples.

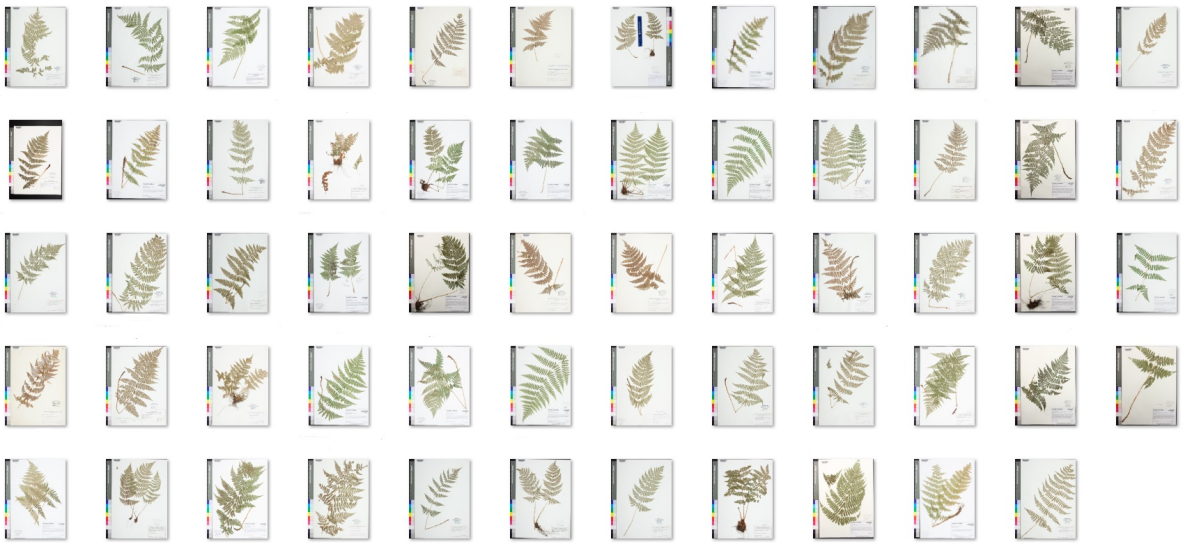

Figure 5.28: BOONE sample images belonging to the best performing class *Dryopteris Intermedia* in the group of classes which do not posses 100% top-5 accuracy.

5.3.2 GoingDeeper Dataset

It is interesting to note that researchers Carranza-Rojas et al. were able to obtain 79.6% top-1 accuracy and 90.3% top-5 accuracy on their Herbaria1K dataset with a modified InceptionV3 classifier [6]. Meanwhile, a non-modified InceptionV3 classifier trained using the same process with the researcher-provided metadata and class labels, was only able to obtain a top-1 accuracy of 45.36% on the validation set, and a top-5 accuracy of 68.29%. With twice as many average training images per class to learn from (414.73 and 207.13 respectively) it was predicted that the GoingDeeper dataset would prove an easier classification problem than the Herbaria1K dataset. However, the significant discrepancy of 34.26% top-1 accuracy and 21.81% top-5 accuracy between winning classifiers trained on these datasets would suggest that this is not the case.

An additional observation worth mentioning, is the notable difference in performance between the best performing BOONE model resulting from the grid search, and

the best performing GoingDeeper dataset model. The winning BOONE model obtained 72.48% top-1 accuracy and 94.18% top-5 classification accuracy on a dataset containing 88 class labels. The winning GoingDeeper model obtained 45.36% top-1 accuracy on the validation set, and a top-5 accuracy of 68.29% on a dataset 851 class labels. Interestingly enough, for the BOONE dataset there was a mean number of 28 samples-per-class. For the GoingDeeper dataset there was a mean number of 414.73 images per class. It was initially hypothesized that the classifier with more sample images per class would perform better, despite the challenge of having more classes to distinguish between.

Chapter 6

Conclusion

6.1 Conclusion

6.1.1 Examining Feasibility of Classification

Based on the results obtained on the BOONE dataset it appears feasible to perform automated species identification within Appalachian State University's I.W. Carpenter, Jr. Herbarium. If the classifier only issues predictions for class labels whose top-1 positive predictive value is at or above a threshold of 95%, then 137 of 447 of all specimen images present in the validation dataset can be classified automatically. The other 310 specimens present in the validation set would require some degree of user interaction to classify reliably. Among them 288 have the correct class label appear among the top-5 predictions. This leaves 22 remaining samples that require a manual labeling process. In conjunction, these two techniques are capable of classifying about 31% of specimen images automatically, about 64% from a drop-down of five potential classes, and the

remaining 5% would require a manual transcription. However, since hyperparameter tuning was performed on the validation dataset it is anticipated that this performance will degrade on unseen data.

The results obtained on the GoingDeeper dataset suggest that it may not be feasible to perform automated species identification on such a massive scale across so many unique institutions, with only a single neural network based classifier. Hence, the results of this experiment failed to validate the findings of Carranza-Rojas et al. [6]. If the classifier only issues predictions for class labels whose top-1 positive predictive value is at or above a threshold of 95%, then 42 of all 56,470 specimen images present in the validation dataset can be classified automatically. The other 56,428 specimens present in the validation set would require some degree of user interaction to classify reliably. Among them 38,620 have the correct class label appear among the top-5 predictions. This leaves 17,808 remaining samples that require a manual labeling process. Combined, these two techniques are capable of classifying about .1% of specimen images automatically, about 68.4% from a drop-down menu of five potential classes, and the remaining 31.5% would require a manual transcription. Since hyperparameter tuning was performed on the validation dataset, it is anticipated that this performance will degrade on unseen data.

6.1.2 Exploration of a Candidate Hyperparameter Space

In accordance with the results obtained during the exploration of the chosen candidate hyperparameter space, it can be ascertained that there was not a single optimal hy-

perparameter subset which spanned both datasets. The only common hyperparameter identified across both datasets was the weight initialization technique of He Normal initialization. This supports existing machine learning literature which suggests that He Initialization is superior to initialization from the normal distribution, in deep neural networks [30]. Beyond the choice in weight initialization technique, a shared hyperparameter set may still exist across both datasets, but such a hyperparameter set was not identified by this work. It is anticipated that future experiments utilizing these datasets may still require individual hyperparameter tuning. If this is indeed the case, then regrettably, future experimenters will have to incur the associated massive computational overhead involved with this process.

A final observation is that the winning hyperparameter set of the grid search conducted on the BOONE dataset performed significantly differently when applied to the GoingDeeper dataset. When the winning hyperparameters of the BOONE grid search were utilized on the GoingDeeper dataset, a top-1 accuracy of 37.68% was obtained, alongside a top-5 accuracy of 58.75%. Interestingly, the loss function when operating under Adam-based optimization, on the GoingDeeper dataset, failed to converge and early stopping was invoked on epoch 70 after a training time of 21 minutes and 17 seconds. The reason why Adam optimization failed to converge whereas Nesterov-based optimization did not, is a topic which merits future investigation.

6.2 Closing Remarks

Within this body of work, the following have been accomplished. First, a review of existing literature regarding automated plant identification was conducted and analyzed. Difficulties associated with the problem, such as morphologically similar species, the taxonomic impediment, and the species problem, were subsequently identified. Historical attempts at automating this process were examined, and the technique of convolutional neural networks was identified as the most successful of past approaches. Then the recent research of Carranza-Rojas et al. [6] was summarized, and their metadata obtained as a performance benchmark. An experimental setup was conceived to replicate the results of their work as best possible.

Secondly, two data collection pipelines were envisioned and implemented, which utilized multi-threaded web scrapers to obtain all available specimen images for both relevant datasets. Then data cleaning and class name resolution was performed, with a history of the programmatic modifications made to the data retained for user verification. After this, feature extraction was performed for all images via forward propagation through a TensorFlow-based InceptionV3 classifier, with the resulting bottleneck vectors undergoing serialization for performance purposes.

Afterword, an experiment was performed which attempted to identify a common set of hyperparameters for use across both obtained datasets. A total of 96 neural network models were trained, 48 for each dataset. A grid search was run on both datasets, and the winning model was serialized and retained. The candidate hyperparameters for this experiment were carefully selected utilizing recommendations from both: ma-

chine learning literature, and the publicised research material identified during literature review.

The winning models from the grid search were then analyzed and compared to the findings of Carranza-Rojas et al. [6]. Visualizations showcasing the chosen performance metrics of top-1 and top-5 accuracy were developed and discussed. Finally, a conclusion was issued regarding the feasibility of the trained models for classification purposes in this domain. During which, it was stated that such automated approaches are found to be feasible within the confines of the I.W. Carpenter Jr. Herbarium, but may not be feasible across multiple herbaria (as in the GoingDeeper dataset). Final contributions of this work, which may be found published alongside this document online, include:

- publication of all source code, including that which was utilized to perform sklearn-based grid searches in TensorFlow
- mappings between raw and clean class labels for the benefit of future researchers
- publication of the cleaned datasets and associated bottleneck vectors for ease of replicability
- serialized trained classifiers from each dataset ready for use

Bibliography

[1] New York Botanical Garden's Virtual Herbarium, "Index herbariorum: A global directory of public herbaria and associated staff." World Wide Web electronic publication, http://sweetgum.nybg.org/science/ih/. (accessed 2018).

[2] National Resource for Advancing Digitization of Biodiversity Collections, "iDigBio integrated digitized biocollections." World Wide Web electronic publication, https://www.idigbio.org/. (accessed 2019).

[3] D. P. Bebber, M. A. Carine, J. R. Wood, A. H. Wortley, D. J. Harris, G. T. Prance, G. Davidse, J. Paige, T. D. Pennington, N. K. Robson, and R. W. Scotland, "Herbaria are a major frontier for species discovery," *PNAS*, vol. 107, no. 51, 2010.

[4] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*. New Jersey, United States: Pearson Education, Inc., 3rd ed., 2010.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[6] J. Carranza-Rojas, H. Goëau, P. Bonnet, E. Mata-Montero, and A. Joly, "Going deeper in the automated identification of herbarium specimens," *BMC EVOL BIOL*, vol. 17, 2017.

[7] M. Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228–235, 2017.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[9] M. H. Mahmud and S. R. Ray, "Transfer learning using Kolmogorov complexity: Basic theory and empirical evaluations," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS, (USA), pp. 985–992, Curran Associates Inc., 2007.

[10] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *arXiv e-prints*, p. arXiv:1808.01974, 2018.

[11] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 2 of *NIPS*, (Cambridge, MA, USA), pp. 3320–3328, MIT Press, 2014.

[12] C. Gaston and M. O'Neill, "Automated species identification: Why not?," *Phi. Trans. R. Soc. Lond. B Bio. Sci.*, vol. 359, no. 1444, pp. 655–670, 2004.

[13] M. W. Ellis, "The problem with the species problem," *HPLS*, vol. 33, no. 3, pp. 343–363, 2011.

[14] J. Wieczorek, D. Bloom, R. Guralnick, S. Blum, M. Döring, R. Giovanni, T. Robertson, and D. Vieglais, "Darwin core: An evolving community-developed biodiversity data standard," *PLOS ONE*, vol. 7, no. 1, pp. 1 - 8, 2012.

[15] B. I. Standards, "Darwin Core quick reference guide." World Wide Web electronic publication, https://dwc.tdwg.org/terms/#dwc:scientificName. (accessed 2018).

[16] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Phil. Trans. R. Soc. A*, vol. 374, 2016.

[17] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[18] C. R. Rao, "The utilization of multiple measurements in problems of biological classification," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10, no. 2, pp. 159–203, 1948.

[19] S. M. Lucas, "Continuous n-tuple classifier and its application to real-time face recognition," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 145, pp. 343–348(5), 1998.

[20] L. Boddy, C. W. Morris, M. F. Wilkins, L. Al-Haddad, G. A. Tarran, R. R. Jonker, and P. H. Burkill, "Identification of 72 phytoplankton species by radial basis function neural network analysis of flow cytometric data," *MEPS*, vol. 195, pp. 47–59, 2000.

[21] D. Wijesingha and F. Marikar, "Automatic detection system for the identification of plants using herbarium specimen images," *Tropical Agricultural Research*, vol. 23, 2012.

[22] C. L. E. Forum, "PlantCLEF 2015." World Wide Web electronic publication, http://www.imageclef.org/lifeclef/2015/plant. (accessed 2018).

[23] J. Yangqing, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint*, p. arXiv:1408.5093, 2014.

[24] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Dokl. Akad. Nauk SSSR*, vol. 163, pp. 845–848, 1965.

[25] W. McKinney, "Data structures for statistical computing in Python"," in *Proceedings of the 9th Python in Science Conference* (S. Walt and J. Millman, eds.), pp. 51–56, 2010.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[27] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. North Sebastopol, CA: O'Reilly Media, Inc., 1st ed., 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv e-prints*, p. arXiv:1412.6980, 2014.

[29] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 2010.

[30] S. K. Kumar, "On weight initialization in deep neural networks," *CoRR*, vol. abs/1704.08863, 2017.

[31] S. Raschka, *Python Machine Learning*. Birmingham, UK: Packt Publishing Ltd., 1st ed., 2016.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, pp. 1–9, 2015.

Vita

Christopher Leigh Campell was born in Durham, North Carolina, to Robert Campell and Diana Meisburger in May of 1994. As a high school student, Chris developed a passion for computers while he volunteered at the Kramden Institute, learning how to disassemble and triage donated machines. Chris graduated from Durham School of the Arts in 2012 with a focus in the digital media arts (the closest his school offered to computer programming). The following autumn, Chris entered Appalachian State University to study his high school passion of computer science. While pursuing his B.S. degree, Chris elected to spend an additional fifth year as an undergraduate senior to explore a variety of CS related career paths. During his exploratory fifth year of studies, he discovered his passion for machine learning and artificial intelligence, and began to take independent course work on the subject manner sourced from Harvard University. Simultaneously, Chris began to work on his M.S. degree in computer science, with the intent of turning his newly found passion for machine learning into a career. During the same year, while finishing up his undergraduate studies, Chris was selected to present machine learning research performed for his undergraduate senior capstone at the National Conference on Undergraduate Research in Memphis Tennessee.

Meanwhile, Chris gained employment experience working as a Student Computer Consultant with Appalachian State University's Information Technology Support (ITS) services. Simultaneously, Chris explored life as a system administrator while employed as a Technology Support Technician with Appalachian State's Technology Support Services (TSS). Chris

was awarded a B.S. degree in Computer Science in August 2017. The following year, during his graduate studies, Chris accepted both a research assistantship with the Appalachian State University Biology Department, and a teaching assistantship with the Computer Science Department. Since this time, Chris has served as the sole teaching assistant for newly offered courses on machine learning, artificial neural networks, and data mining; despite being largely self-taught in the subject matter. Additionally, Chris has served as a teaching assistant for interdisciplinary courses offered on data programming and data visualization, during which he has enjoyed improving upon his abilities to explain complex topics to non-computer science majors. In September of 2019 Chris joined the company MachNation as a Technology Analyst who specializes in machine learning and data analysis. Chris was awarded his M.S. in computer science in December 2019. He now resides in Raleigh, North Carolina, where he plans to follow his dream by continuing a career in machine learning and/or artificial intelligence.